5 Appendices

5.1 Determination of tri-valent titanium (Ti^{3+}) in titania slag

Calculations

The principle behind the determination of the trivalent titanium is the formation of reduced forms of titanium and iron when leached with ferric sulphate. The ferrous sulphate formed during the reduction of ferric ammonium sulphate is determined by titration with 0.1N potassium permanganate using 1.10-phenanthroline as an indicator. Iron will form iron (II) and therefore a correction must be made in the determination of Ti^{3}+.

Equations

$$Fe^{0} + 2Fe^{3+} \rightarrow 3Fe^{2+}$$

$$Ti^{3+} + Fe^{3+} \rightarrow Ti^{4+} + Fe^{2+}$$

$$\%Ti_2O_3 \text{ (expression in terms of } TiO_2) = \left[0.00799(A-B)\frac{100}{W}\right] - \left[1.43(C+2D)\right]$$

Where A = Titrant value of potassium permanganate solution, cm^3

B = Blank titrant value of potassium permanganate solution, cm^3

C = Fe (tot) (from XRF)

D = Metallic iron content (Determined as per Standard Task Procedure for **Determination of metallic iron (Fe^0) in Titanium Slag**)

W = Mass of the sample (mg)

1.43 = Conversion factor for Fe to Fe_2O_3

79.9 = Molar mass of TiO_2

Allowable variance in Ti^{3+}: 0.5%

Steps	**Operating/Quality Criteria**			✓×	**Hazard and Risk Exposure**	**Control Measures**	✓×
Check equipment/ tools/chemicals	Analytical balance (4 decimal place)		Titrasol ampoule (0.1N $KMnO_4$)		Do not have the correct equipment, tools or chemicals	Obtain new or safe replacement before commencing with the task.	
	Water purifier		Stop watch				
	Magnetic stirrer bars		Stainless sink				
	Stirrer rod		Funnel, glass				
	Indicator bottle with a medicine dropper		Measuring cylinders, glass (25ml, 100ml, 500ml, 1000ml)				
	Volumetric flask, (100ml, 1000ml)		Measuring cylinder, plastic (10ml)				
	Hotplate with sand bath		Erlenmeyer flask				
	Magnetic stirrer/hotplate		1.1 Phenanthroline				
	Wash bottle		Sulphuric Acid (95-99%)				
	Beaker, teflon (250ml)		Boric Acid crystals (AR)				
	Beaker, glass (50ml, 250ml, 1000ml)		Calcium Gluconate Solution				
	Hydrofluoric Acid, HF (48%)		Ammonium Ferric Sulphate				
	Burette (50ml)		Ferrous sulphate				

Steps	Operating/Quality Criteria			✓×	Hazard and Risk Exposure	Control Measures	✓×
	CRM (SARM 57)						
1. Check Personal Protective Clothing (PPC)	Overalls/Lab coat		Safety shoes		Do not have the correct PPC	Obtain the appropriate PPC before commencing with the task	
2. Check Personnel Protective Equipment (PPE)	Nitrile gloves		Safety glasses		Do not have the correct PPE	Obtain the appropriate PPE before commencing with the ask	
	Latex gloves		Heat resistant gloves				
4. Ensure following documentation is available	Standard Task Procedure for: Calibration of, and weighing on an Analytical Balance (Powerdocs: IHM_CPC# 26953).				Documentation is not accessible from the computerized system.	Place a breakdown log with AST Help Desk, Dial 5000	
	Wet Chemistry Daily Log sheet (Powerdocs: IHM_CPC#47800).						
	Standard Task Procedure for: Cleaning up of chemical spillages (Powerdocs: IHM_CPC#27097).				Documentation is not accessible from the computerized system.	Place a breakdown log with AST Help Desk, Dial 5000	
	Standard Task Procedure for: Waste Management- Hydrogen Fluoride (HF) (Powerdocs: IHM_CPC#60720)						
	Standard Task Procedure for: Waste Management- Altered, Unaltered and Slag waste (Powerdocs: IHM_CPC#60726)						

Steps	Operating/Quality Criteria	✓×	Hazard and Risk Exposure	Control Measures	✓×
	Standard Task Procedure for Waste management – Glassware (Powerdocs: IHM_CPC#60724)				
5 Procedure	Inspect all glassware for chips or cracks before and after use.		Injury to hands	Use nitrile gloves	
			Incorrect disposal	Dispose of broken glassware into the “Broken glassware bin”.	
5.1 Preparation of reagents	Ensure that the balance has been calibrated as per Standard Task Procedure for **Calibration of, and weighing on an analytical Balance.**		Not calibrated	Calibrate and verify the balance before commencing with task.	
	Reagents are prepared using only “A” grade calibrated glassware.				
a) **Potassium permanganate solution** ($KMnO_4$) 0.1N	Prepare a titrasol ampoule of 0.1N $KMnO_4$ in a 1000ml volumetric flask.				
	Carefully break off the neck of the ampoule using the blade supplied., or twist off the neck (depending on which container is supplied).		Injury to hands.	Wear nitrile gloves.	
	Quantitatively transfer the contents of the ampoule into a clean dry 1000ml volumetric flask using a funnel.		Spillage	Discard and reprepare the reagent.	
	Rinse the neck and ampoule into the flask, at least 3 times with distilled water from a wash bottle.		Incorrect normality	Ensure that every drop is washed into the flask.	
				Rinse down the funnel as well.	
	Make up to the mark with distilled water.		Over shot the mark	Discard and remake the solution.	

Steps	Operating/Quality Criteria	✓×	Hazard and Risk Exposure	Control Measures	✓×
	Homogenise and transfer the solution into a dark coloured reagent bottle.				
	Seal, and label the reagent bottle with the name of reagent, date, time and name of analyst.		Incorrect labelling.	Ensure that the correct labelling is done.	
b) **Sulphuric Acid Solution** 50 % (1:1) (Stock Solution)	Quantitatively transfer 450ml distilled water into a clean dry 1000ml glass beaker using a 500ml measuring cylinder.		Water purifier not working	Replace DI cartridges.	
	Place the 1000ml beaker into the stainless sink.				
	Plug the sink using the stopper and fill with cold water to create a water bath.		Beaker begins to float and tips over.	Fill sufficient water in the bath to allow the beaker to be immersed without floating.	
	Quantitatively transfer 500ml of Sulphuric acid (95-99%) into a clean dry 500ml measuring cylinder		Inhalation due to fumes	Work in a fume hood.	
			Injury to hands	Wear nitrile gloves.	
			Spillage	Adhere to the Standard Task Procedure for **Cleaning up of chemical spillages.**	
	Slowly transfer small quantities at a time of the 500ml Sulphuric acid into the 1000ml beaker containing the 450ml of distilled water using a glass funnel.		Solution heats rapidly.	Add the acid slowly. Keep the volumetric flask cool by swirling it in the water bath or stir with a stirrer rod.	
				Let the H_2SO_4 run onto the side of the beaker.	
				Always add acid to water.	

Steps	Operating/Quality Criteria	✓✗	Hazard and Risk Exposure	Control Measures	✓✗
			Injury to hands	Use nitrile gloves	
	Allow the solution to cool to room temperature in the water bath.				
	Once cooled, transfer to a clean dry 1000ml volumetric flask using a funnel and make up to the mark with distilled water.		Mark is over shot	Discard and reprepare.	
	This is now the 50% sulphuric acid stock solution.				
	Transfer to a reagent bottle, seal, and label the reagent bottle with the name of reagent, date, time and name of the laboratory technician.		Incorrect labelling.	Ensure that the correct labelling is done.	
	Transfer the stock solution to the designated auto dispenser when required.		Contamination of reagent	Ensure that the designated dispenser is used.	
				Rinse the dispenser with the new stock solution before filling up.	
c) **Boric Acid Solution** (H_3BO_3)	Add ~500ml of distilled water into a clean 2000ml volumetric flask.				
	Weigh 48g ± 0.01g of dry Boric acid (H_3BO_3) crystals into a clean dry 50ml beaker.		Incorrect sample weight.	Discard and reweigh.	
	Place a clean dry glass funnel into the 2000ml volumetric flask containing the distilled water.				
	Quantitatively transfer the boric acid crystals into the volumetric flask by rinsing out the beaker with ~1000ml distilled water over the glass funnel.		Substances sticking to sides of the glassware	Ensure the sides of the glassware are washed down with water.	

Steps	Operating/Quality Criteria	✓×	Hazard and Risk Exposure	Control Measures	✓×
	Quantitatively transfer 220ml of Sulphuric acid (95-99%) into a clean dry 500ml measuring cylinder.		Inhalation due to fumes	Work in a fume hood.	
			Injury to hands	Wear nitrile gloves.	
			Spillage	Adhere to the Standard Task Procedure for **Cleaning up of chemical spillages.**	
	Slowly transfer small quantities at a time of the 220ml sulphuric acid into the 2000ml volumetric flask containing the boric acid solution.		Solution heats rapidly.	Add the acid slowly. Continuously swirl the flask under running cold water.	
			Injury to hands	Wear nitrile gloves.	
	Allow the solution to cool.				
	Once cooled, make up to the mark with distilled water.				
	Transfer to a reagent bottle, seal, and label the reagent bottle with the name of reagent, date, time and name of the laboratory technician.		Incorrect labelling.	Ensure that the correct labelling is done.	
	Transfer to the designated auto dispenser when required.		Contamination of reagent	Ensure that the designated dispenser is used.	
				Ensure that the designated dispenser is cleaned and rinsed with the new stock solution.	
d) **Ammonium Ferric Sulphate** (Stock Solution)	Weigh out accurately 25g ± 0.01g of Ammonium Ferric Sulphate into a clean dry 250ml beaker.		Incorrect sample weight.	Discard and reweigh.	

Steps	Operating/Quality Criteria	✓×	Hazard and Risk Exposure	Control Measures	✓×
	Using a measuring cylinder, add 150mls of distilled water into the beaker.		Water purifier not working	Replace DI cartridges.	
	Quantitatively transfer 75ml concentrated Sulphuric Acid (95-99%) into a 100ml measuring cylinder		Injury to hands	Wear nitrile gloves.	
			Inhalation due to fumes	Work in the fume hood.	
	Carefully add the 75ml Sulphuric Acid to the beaker containing the Ammonium Ferric Sulphate.		Solution gets hot	Allow the cool water to flow under the beaker.	
	Mix the solution well using a stirring rod.		Injury to hands	Wear nitrile gloves	
	Allow the solution to cool to room temperature.		Cooling takes too long	Force cool in a water bath.	
	Transfer the solution to the designated dark coloured reagent bottle. This is now the ammonium ferric sulphate stock solution.		Contamination of reagent	Ensure that the designated reagent bottle is used.	
				Ensure that the designated reagent bottle is cleaned and rinsed with the new stock solution.	
			Reagent is sensitive to light.	Store in a dark bottle, inside a cupboard.	
	Seal, and label the reagent bottle with the name of reagent, date, time and name of analyst.		Incorrect identification	Ensure the correct labelling is done.	
e) Phenantholine indicator	Weigh out accurately, 0.7g ± 0.01g Ferrous sulphate into a clean dry weighing boat.		Incorrect sample weight.	Discard and reweigh the sample.	
	Weigh 1.0g ± 0.01g of 1.1 Phenantholine into a separate clean dry weighing boat.		Incorrect sample weight.	Discard and reweigh the sample.	

Steps	Operating/Quality Criteria	✓✗	Hazard and Risk Exposure	Control Measures	✓✗
	Place a clean dry glass funnel into a 100ml volumetric.				
	Quantitatively transfer the chemicals into the volumetric flask by rinsing out the beaker with distilled water over the glass funnel.		Substances sticking to sides of the weighing boats.	Ensure the sides of the weighing boats are washed down with water.	
	Make up to the mark with distilled water.				
	Homogenize the solution.		Spillages.	Hold the flask firmly.	
				Ensure the cap is on tightly.	
	Transfer to a reagent bottle, seal, and label the reagent bottle with the name of reagent, date, time and name of the laboratory technician.		Incorrect labelling.	Ensure that the correct labelling is done.	
5.2 Quality Control	A Quality control sample is prepared and analysed as per section 5.3, in duplicate with every batch of unknown samples.		Quality sample is out of limits	Do not report the production results	
				Repeat the analysis.	
	Perform the analysis on LIMS by selecting the appropriate sample ID from LIMS **SPC Ti_2O_3 & Fe (Metallic)** and the method T**rivalent (Ti^{3+}) Sarm57**		LIMS not available	Record the sample ID on the Wet chemistry Daily Log sheet.	
	SARM 57 is analysed in duplicate as per section 5.3. The average is reported.				
5.3 Preparation of the blank and sample	Unknown samples are prepared and analysed in duplicate and the average is reported.				

Steps	Operating/Quality Criteria	✓×	Hazard and Risk Exposure	Control Measures	✓×
	A blank should be prepared when new reagents are prepared and analysed with each batch of production samples.				
	Accurately weigh out 0.5000g ± 0.0010g of milled sample into a weighing boat.		Incorrect sample weight.	Discard and reweigh.	
				For the preparations of a blank, no sample is added.	
	Quantitatively transfer the weighed sample into a teflon beaker.				
	Add 15ml of the ammonium ferric sulphate stock solution into a teflon beaker, using a 25ml measuring cylinder.				
	Swirl to cover the entire sample.		Injury to hands.	Wear nitrile gloves.	
			Injury to eyes.	Wear safety glasses.	
	Transfer 20ml of 50% sulphuric acid into the teflon beaker, using the dispenser.		Incorrect volume dispensed	Ensure that the desired volume setting on the dispenser is selected before dispensing the solution.	
	Place the teflon beaker on the sand bath of the hot plate.				
	Bring to boil. Allow boiling for 1 minute while swirling by timing with a stopwatch.		1 minute exceeded	Repeat the preparation from step 5.3	
	Quantitatively transfer 10ml HF into 10ml plastic measuring cylinder, or using the dispenser, dispense 10 ml HF directly in to Teflon beaker.		Injury to hands	Use latex/nitrile gloves.	
			Inhalation due to fumes	Work in the fume hood.	

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Steps	Operating/Quality Criteria	✓✗	Hazard and Risk Exposure	Control Measures	✓✗
			Damage to glassware	Use a plastic measuring because HF attacks the silica in glass.	
			Skin Contact	Avoid contact with skin. Wash with copious amount of water and then apply calcium gluconate solution to the area. Calcium gluconate solution is kept in the fridge in the Wet Chemistry Laboratory.	
				Seek medical help.	
	Add the 10ml HF into the teflon beaker.				
	Place the teflon beaker on the hotplate and boil at a lower heat setting for 10 minutes while swirling.				
	After boiling for 10 minutes remove the teflon beaker from hot plate.		Injury to hands.	Use heat resistant/nitrile gloves.	
	Immediately add 100ml boric acid stock solution to the teflon beaker using the dispenser.		Inhalation due to fumes.	Work in the fumehood.	
	Cool to room temperature.				
	Add 10 drops of the Phenanthroline indicator into the teflon beaker, using the medicine dropper.				
	Place a magnetic stirrer bar into the teflon beaker and stir well on the magnetic stirrer.				

Steps	Operating/Quality Criteria	✓×	Hazard and Risk Exposure	Control Measures	✓×
5.4 Titration of the sample	Rinse a clean dry 50ml burette out with 0.1N KMnO4 solution.		Contamination	Rinse the burette with the solution to ensure it is free from contaminants.	
	Fill the 50ml burette to the mark with 0.1N $KMnO_4$.				
	Add drop wise $KMnO_4$ until the endpoint is reached.		Endpoint is exceeded	Repeat the analysis from step 5.3	
				Endpoint is reached when solution changes to a pale blue colour.	
	Enter the titration value (volume of the 0.1N $KMnO_4$used), on LIMS under the appropriate sample type and method		Incorrect LIMS method selected.	Select the appropriate sample ID and the method T**rivalent (Ti^{3+}) Sarm57** for the control sample and **Trivalent titanium (Ti^{3+})** for production samples.	
			LIMS is not accessible	Record the titrant value (volume of the 0.1N $KMnO_4$ used) on the Wet Chemistry Daily Log sheet	
5 Reporting of results	All calculations are performed by LIMS				
	Quality control samples are analysed in duplicate and the average is reported.				
	Unknown samples are analysed in duplicate and the average is reported.				
7 Waste Management	Discard liquid mercury chloride waste and contaminated waste paper in the designated mercury chloride waste bins in the Wet Chemistry Laboratory.		Incorrect waste disposal.	Adhere to Standard Task Procedure for **Waste Management: Mercury Chloride**.	

Steps	Operating/Quality Criteria	✓✗	Hazard and Risk Exposure	Control Measures	✓✗
	Discard altered slag waste in the appropriate bins in the mill and pressroom.		Incorrect waste disposal	Adhere to Standard Task Procedure for **Waste Management: Altered, Unaltered and Slag Waste**	
	Discard chipped or broken glassware into the appropriate bin in the Wet Chemistry Laboratory.		Incorrect waste disposal	Adhere to Standard Task Procedure for **Waste Management: Glassware**	
8 Housekeeping	Wash all glassware and store away.		Damage to glassware.	Ensure that all damage to equipment/glassware is reported before the end of the shift.	
	Clean the workbench on completion of analysis.				
	Return all reagents to their respective storage cupboards				
	Clean up all spillages		Spillage	Adhere to the standard task procedure for **Cleaning up of chemical spillages.**	

5.2 List of parameters evaluated in the search for parameters correlating with the -106 µm fraction.

initial block mass

block yield

yield mass

%FeO

$\%TiO_2$

$\%Al_2O_3$

%CaO

$\%Cr_2O_3$

%MgO

%MnO

$\%SiO_2$

$\%V_2O_5$

primary cooling time (hrs)

Tapping rate (kg/min)

Tapping rate$^{-0.5957}$ (kg/min)

Equivalent %FeO

(Equivalent %FeO)^2

$\%Ti_2O_3$

$\%TiO_{2\ true}$

$(\%TiO_{2\ true})^2$

Equivalent $\%Ti_2O_3$

(Equivalent $\%Ti_2O_3)^2$

$SiO_2+Al_2O_3$(glass)+CaO

Specific surface area - initial

Ln(spec surface area - initial)

Specific surface area - after

Ln(spec surface area - after)

(Specific surface area - after)$^{3.6321}$

$T_{liquidus}$ (calculated)

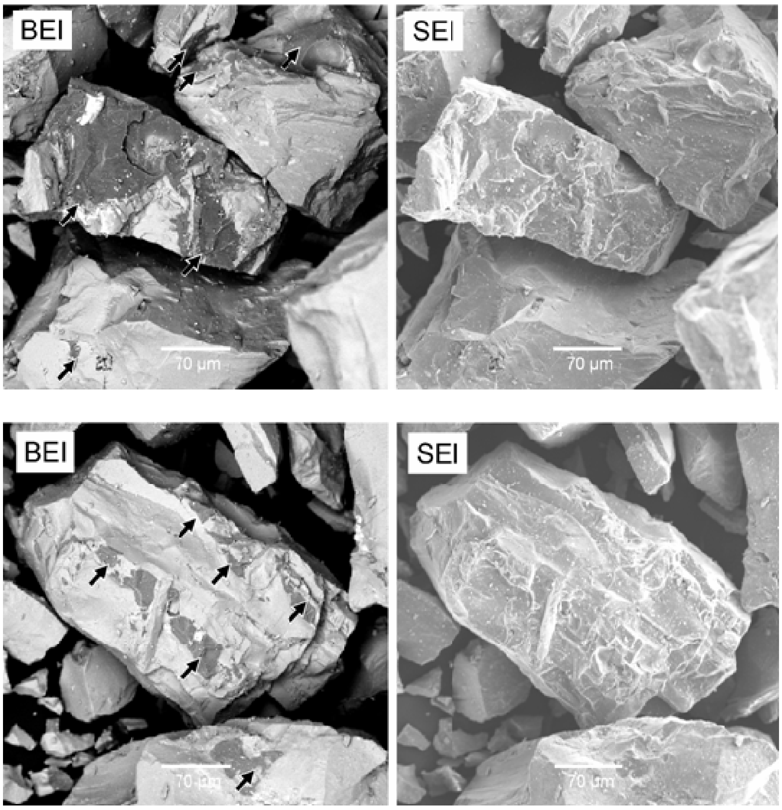
$T_{solidus}$ (calculated)

T_{tap}

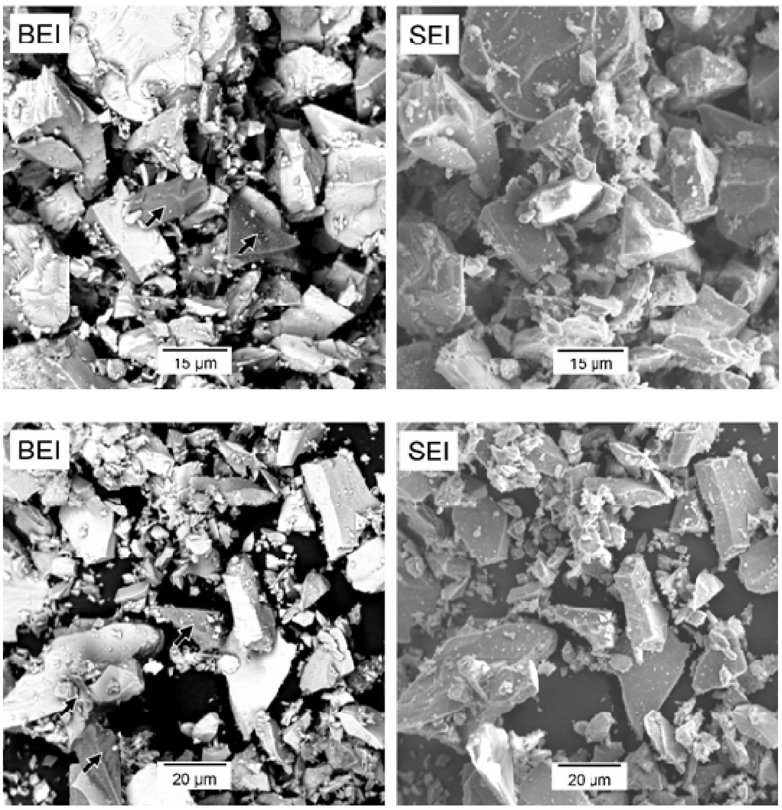
superheat

5.3 SEM examples of SiO_2 on particle surfaces

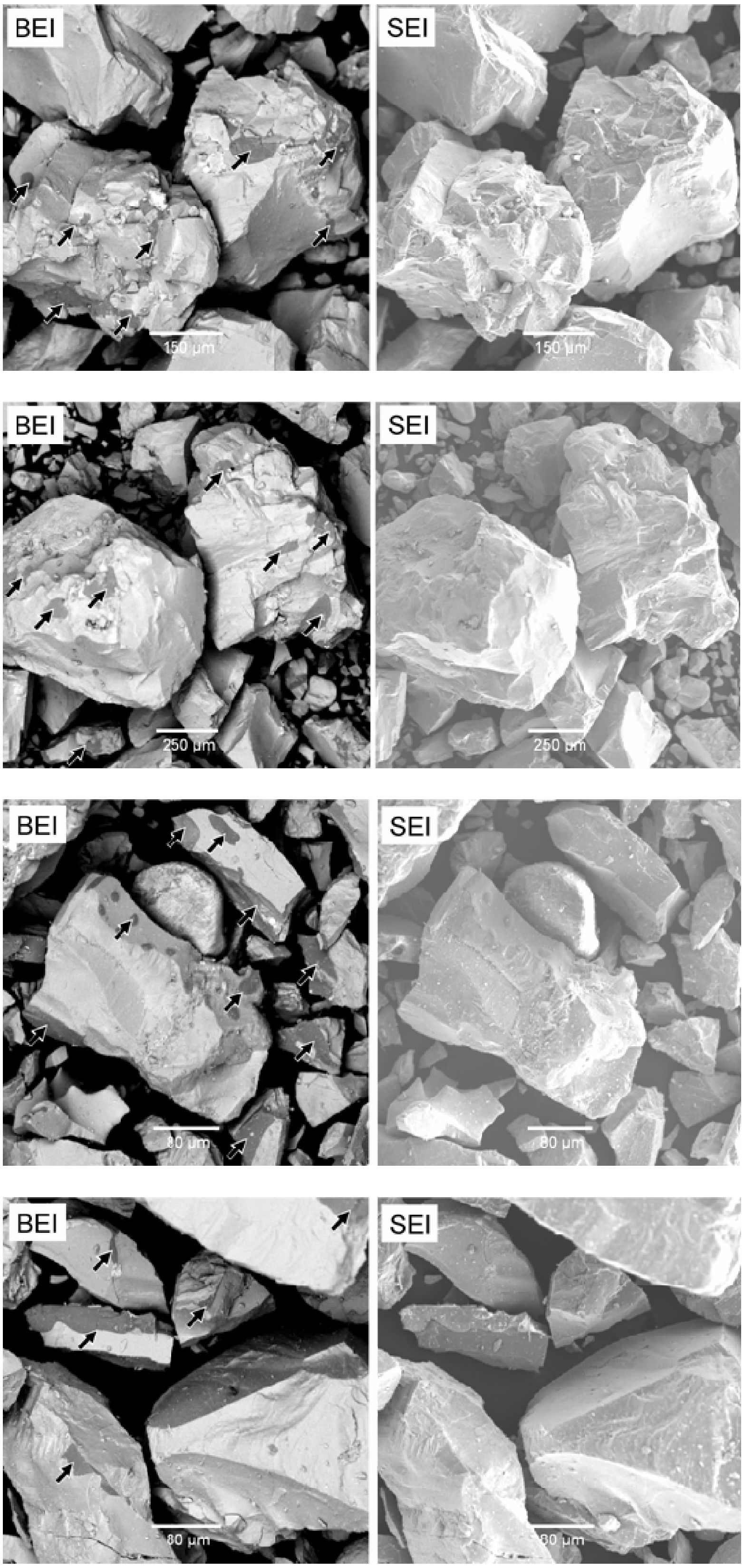
Coarse fraction (+850 µm -106 µm)



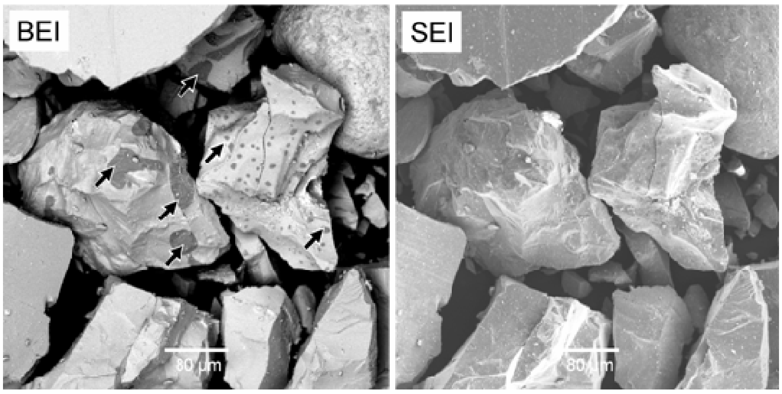
Fine fraction (-106 µm)



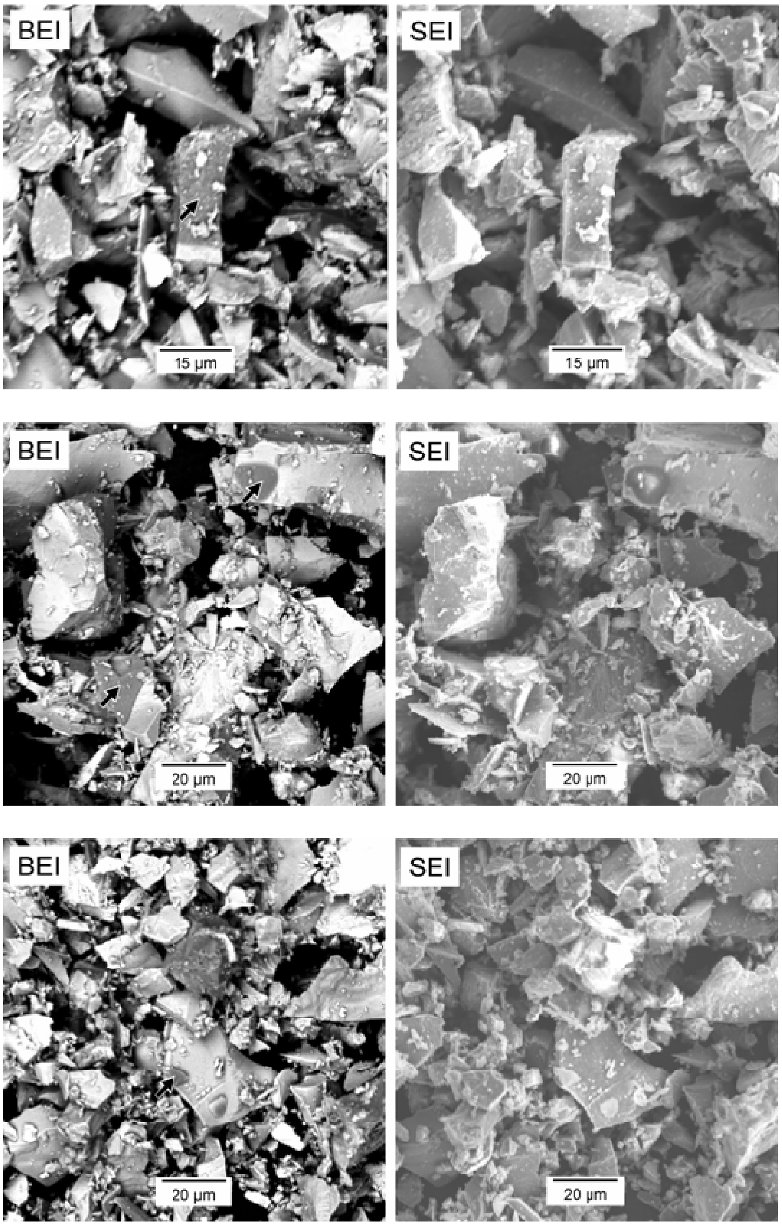
Coarse fraction



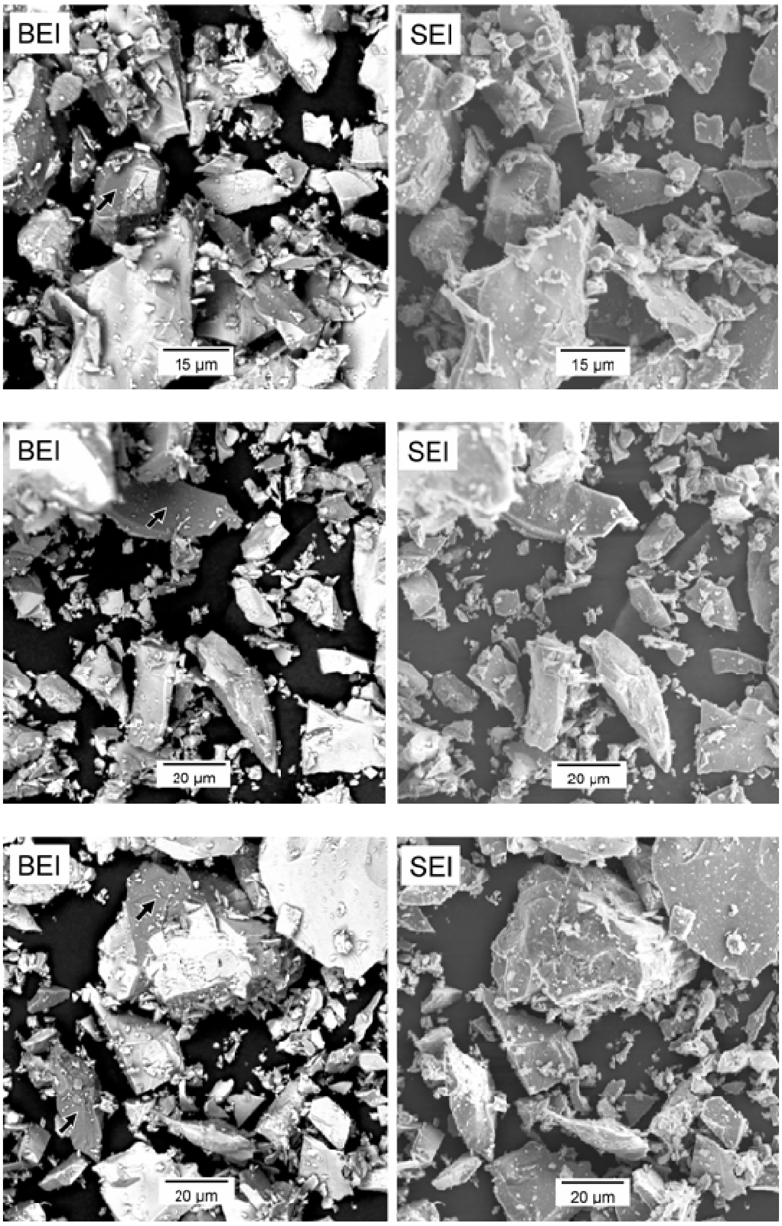
Coarse fraction



Fine fraction



Fine fraction



5.4 FlexPDE code for the one dimensional example

```
! ########################################################################################
!  Neumann problem.pde
!
!       Date:           2007-10-28
!       Developer:      Hanlie Kotze
!       Model Purpose:  Part of PhD studies.  Objective is to have a numerical and analytical solutions of
a semi-infinite volume which starts off at a temperature above it's liquidus and cools down to
solidiifcation.
!       The shell thickness as a function of time, and temperature profile at a given time, are compared
for the two solutions.  The analytical solution was done by Prof Chris Pistorius - "Neumann problem.xls".
Equations are from Carslaw & Jaeger, 1959.  The following is the numerical solution.
!       Units:          Temperature             [°C]
!                       Distance        [m]
!                       Time                    [s]
!                       Density         [kg/m3]
!                       Mass                    [kg]
!                       Heat Capacity           [J/(kg.°C)]
!                       Thermal Conductivity    [W/(m.°C)]
!       References:
!       1       Carslaw & Jaeger, 1959
!       Assumptions:
!       1       Material properties are constant (not a function of temperature)
!       2
!
! ########################################################################################

TITLE
        'Neumann problem'

COORDINATES
        CARTESIAN1

SELECT
        SMOOTHINIT
        ERRLIM = 1E-8
        NGRID = 10
        NODELIMIT = 100

VARIABLES
        temperature

 DEFINITIONS
        ! ####################################################################################
        ! Physical Properties
        ! ####################################################################################
        Cpsolid = 900                           ! heat capacity of the solid [J/kgK]
        Cpliquid = 1000                 ! heat capacity of the liquid [J/kgK]
        Hf = 650000                             ! heat of fusion [J/kg]
        ksolid = 2                              ! thermal conductivity of the solid [W/mK]
        kliquid = 4                             ! thermal conductivity of the liquid [W/mK]
        density = 4000                  ! material density [kg/m3]

        ! #####################################################################################
        ! Boundary Conditions
        ! #####################################################################################
        Tsolidus = 1510
        Tliquidus = 1530
        Tinitial = 1550
        Tsurface = 100
        Maxtime = 10*24*3600
        Domainlength = 2.0
        Fliquid = max( min((temperature - Tsolidus) / (Tliquidus - Tsolidus),1.0), 0)
        Fsolid = max( min((Tliquidus - temperature) / (Tliquidus - Tsolidus), 1), 0)

        Cp =    Cpsolid + SWAGE(temperature-Tsolidus,0,(Cpliquid-Cpsolid)/2+Hf/(Tliquidus-
Tsolidus),(Tliquidus - Tsolidus)/10) - SWAGE(temperature-Tliquidus,0,Hf/(Tliquidus-Tsolidus)+(Cpsolid-
Cpliquid)/2,(Tliquidus - Tsolidus)/10)

        k =     SWAGE(temperature-Tsolidus, ksolid, kliquid, (Tliquidus-Tsolidus))

INITIAL VALUES
        temperature = Tinitial

EQUATIONS
        temperature:    (density*Cp)*dt(temperature) - div(k*grad(temperature)) = 0.0

BOUNDARIES
        REGION 1
                START(0) POINT VALUE(temperature) = Tinitial
                LINE TO (Domainlength) POINT VALUE(temperature) = RAMP(t-50, Tinitial, Tsurface, 500)
        FEATURE 'Line'
                START(0) LINE TO (Domainlength)

TIME
        FROM 0 TO Maxtime
```

```
MONITORS
        FOR t = 0,1,2,4,8,16,32,50 BY 100 TO ENDTIME
        ELEVATION(temperature) ON 'Line'
        ELEVATION(Fliquid, Fsolid) ON 'Line'

PLOTS
        FOR t = 0, 3600, 7200, 14400, 28800, 57600, 115200, 230400, 460800, 921600
        ELEVATION(temperature) ON 'Line' EXPORT FORMAT "#d, #1"
        ELEVATION(Fliquid, Fsolid) ON 'Line'EXPORT FORMAT "#d, #1, #2"

HISTORIES
        HISTORY(temperature) AT (0) AT (Domainlength*.2) AT (Domainlength*.4) AT (Domainlength*.6) AT
(Domainlength*.8) AT (Domainlength*.9) AT (Domainlength-0.1) AT (Domainlength-0.05) AT (Domainlength-0.02)
AT (Domainlength) EXPORT FORMAT "#t #r, #i"

        HISTORY(Fliquid) AT (0) AT (Domainlength*.2) AT (Domainlength*.4) AT (Domainlength*.6) AT
(Domainlength*.8) AT (Domainlength*.9) AT (Domainlength-0.1) AT (Domainlength-0.05) AT (Domainlength-0.02)
AT (Domainlength) EXPORT FORMAT "#t #r, #i"

        HISTORY(Cp) AT (0) AT (Domainlength*.2) AT (Domainlength*.4) AT (Domainlength*.6) AT
(Domainlength*.8) AT (Domainlength*.9) AT (Domainlength-0.1) AT (Domainlength-0.05) AT (Domainlength-0.02)
AT (Domainlength) EXPORT FORMAT "#t #r, #i"

        HISTORY(k) AT (0) AT (Domainlength*.2) AT (Domainlength*.4) AT (Domainlength*.6) AT
(Domainlength*.8) AT (Domainlength*.9) AT (Domainlength-0.1) AT (Domainlength-0.05) AT (Domainlength-0.02)
AT (Domainlength) EXPORT FORMAT "#t #r, #i"

 END
```

5.5 Flex PDE code for the slag block model

File 1: slag block and pot system

```
! ###############################################################################################
!  UP Slag Block Cooling 3a.pde
!
!       Client:           Department of Material Science and Metallurgical Engineering
!                         University of Pretoria
!       Date:             2007-02-13
!
!       Developer:        Johan Zietsman
!
!       Model Type:       Transient
!       Model Purpose:    The model is part of Hanlie Kotze's Ph.D. study. Chris Pistorius requested Ex
Mente to assist with the development of a
!                         model based on Hanlie's formulation. Hanlie would then use this model to
!                         generate results required for the completion of her Ph.D.
!                         The purpose of the model is to describe the cooling and solidification
!                         of a high-TiO2 slag block as it moves through the various cooling stages
!                         and cooling conditions on the plant.
!                         The model must take into account the following factors:
!                         - tap temperature,
!                         - tap composition,
!                         - cooling conditions.
!                         The model must also be able to describe a small pilot-plant configuration
!                         and the full-scale industrial configuration.
!
!       Units:            Temperature               [°C]
!                         Distance          [m]
!                         Time                      [s]
!                         Fraction          []
!                         Chemical Composition      [%]
!                         Density                   [kg/m3]
!                         Mass                      [kg]
!                         Heat Capacity             [J/(kg.°C)]
!                         Thermal Conductivity      [W/(m.°C)]
!
! ###################################################################################################

TITLE
        'UP Slag Block Cooling 6a'

COORDINATES
        YCYLINDER

SELECT
        ERRLIM = 1E-4
        SMOOTHINIT
        GRAY
        NODELIMIT = 1000

VARIABLES
        temperature (1700)

DEFINITIONS
        ! ###########################################################################################
        ! Constants
        ! ###########################################################################################

        g = 9.81          ! Gravitational acceleration      [m/s2]
        sigma = 5.669E-8 ! Stefan-Boltzmann constant        [W/(m2.K4)]
        slagDensity = 3800.0      ! Slag density                    [kg/m3
        steelDensity = 7600       ! Steel density                   [kg/m3]

        ! ###########################################################################################
        ! Model Inputs
        ! ###########################################################################################

!%%%% TAP37
{       slagFeOContent_Analysed = 13.25
        slagTemperature_AtTap = 1669.0
        tapMass = 1002.0}

!%%%%TAP38
{       slagFeOContent_Analysed =13.92
        slagTemperature_AtTap = 1668.0
        tapMass = 1365.0}

!%%%% TAP42 C9 WATERcooling
{       slagFeOContent_Analysed = 11.99
        slagTemperature_AtTap = 1709.0
        tapMass = 1017.0}
```

```
!       %%% Tap 64 C9 AIRcooling
{       slagFeOContent_Analysed = 11.46
        slagTemperature_AtTap = 1683.0
        tapMass = 1420.0}

!       Big tap
        slagFeOContent_Analysed = 10.10
        slagTemperature_AtTap = 1678.0
        tapMass = 18200

        potTemperature_BeforeTap = 25.0
        slagFeOContent = slagFeOContent_Analysed
        slagLiquidusTemperature = (0.2351*slagFeOContent^2 - 11.24*slagFeOContent    + 1664.1)
        slagSolidusTemperature_0 = (0.0364*slagFeOContent^2 - 4.845*slagFeOContent + 1502.7)

        ! Domain Dimension Inputs (1.5 t)
{       potMassCapacity = 1500.0
        potR = 0.523
        potTheta_Degrees = 15.0
        potTheta_Radians = (potTheta_Degrees / 360.0) * 2.0 * PI
        potWallThickness = 0.04}

        ! Domain Dimension Inputs (20 t, dwg 615-BZ00 0002G)
        potMassCapacity = 20000.0
        potR = 1.175
        potTheta_Degrees = 15.0
        potTheta_Radians = (potTheta_Degrees / 360.0) * 2.0 * PI
        potWallThickness = 0.11
        ! ##########################################################################################
        ! Domain Dimensions
        ! ##########################################################################################

        ! Domain Dimension Calculations
        potVolume = potMassCapacity / slagDensity
        blockVolume = tapMass / slagDensity
        potPhi_Degrees = 90.0 - potTheta_Degrees
        potPhi_Radians = (potPhi_Degrees / 360.0) * 2.0 * PI
        blockL = ((potVolume - PI * potR ^ 3.0 / 3.0 / tan(potTheta_Radians)) / (-PI *
(cos(potTheta_Radians)) ^ 3.0 / 3.0 / tan(potTheta_Radians) + PI / 3.0 * (2.0 + cos(potPhi_Radians)) * (1.0
- cos(potPhi_Radians)) ^ 2.0)) ^ (1.0 / 3.0)
        blockrt = blockL*cos(potTheta_Radians)
        blockzt = blockL*(1.0 - sin(potTheta_Radians)) - blockL
        blockVolume_SpericalCap = (PI/3.0) * blockL^3.0 * (2.0 + cos(potPhi_Radians)) * (1.0 -
cos(potPhi_Radians))^2.0
        blockVolume_ConicalSection = (PI/3.0) / tan(potTheta_Radians) * (potR^3.0 - blockrt^3.0)
        blockR = ((3.0*tan(potTheta_Radians)/PI)*(blockVolume - blockVolume_SpericalCap) +
blockrt^3.0)^(1.0/3.0)
        blockZ = (blockR - blockrt)/tan(potTheta_Radians) + blockzt + blockL
        ! ##########################################################################################
        ! Conditions
        ! ##########################################################################################

        ambientTemperature = 25.0

        ! ##########################################################################################
        ! Physical Properties
        ! ##########################################################################################

        k = 0.0          ! Thermal conductivity
        Cp = 0.0 ! Heat capacity

        slagk = (max(0.00175*temperature+0.3,0.3))*1

        slagCp_Solid = - 0.0314*slagFeOContent^2 - 0.4042*slagFeOContent^2 + 908.51
        slagCp_Liquid = 0.0561*SlagFeOContent^2 - 3.3668*slagFeOContent + 1044.3
        slagEnthalpy_At25 = - 139.51*slagFeOContent^2 - 1086.1*slagFeOContent + 515805
        CpSolidify = (slagEnthalpy_At25 + slagCp_Liquid*(slagLiquidusTemperature - 25) -
slagCp_Solid*(slagSolidusTemperature_0 - 25)) / (slagLiquidusTemperature-slagSolidusTemperature_0)

        slagCp=  IF (temperature <= slagSolidusTemperature_0) THEN slagCp_Solid
                        ELSE     IF (temperature >= slagLiquidusTemperature) THEN slagCp_Liquid
                        ELSE (slagEnthalpy_At25 + slagCp_Liquid*(slagLiquidusTemperature - 25) -
slagCp_Solid*(slagSolidusTemperature_0 - 25)) / (slagLiquidusTemperature-slagSolidusTemperature_0)
        slagEnthalpy = IF (temperature <= slagSolidusTemperature_0) THEN slagCp_Solid*(temperature-25)
                        ELSE     IF (temperature >= slagLiquidusTemperature) THEN
slagEnthalpy_At25+slagCp_Liquid*(temperature-25)
                        ELSE     slagCp_Solid*(slagSolidusTemperature_0-25)+CpSolidify*(temperature-
slagSolidusTemperature_0)
        ! Steel
        steelCp = 465.0/slagDensity*steelDensity   ! Carbon Steel 0.5% (Holman)

        steelk = -0.03488*(temperature) + 59.1     ! cast steel, Holman
        ! ##########################################################################################
        ! Initial Conditions
        ! ##########################################################################################

        initialTemperature = 25.0
```

```
        initialLiquidFraction = min(max((temperature - slagSolidusTemperature_0)/( slagLiquidusTemperature
- slagSolidusTemperature_0),0),1)

        !TRANSFER("transfer_1.dat", initialTemperature, initialLiquidFraction)
        ! ##########################################################################################
        ! Boundary Conditions
        ! ##########################################################################################

        ! Water cooling.
        waterVapourDensity = 0.5863                 ! At 107°C (Holman)
        waterLatentHeat = 2256000.0                 ! Latent heat of water to gas reaction, J/kg (internet
data)
        waterSurfaceTension = 0.0588                ! [N/m] (internet data)
        waterDensity = 994.9                        ! At 32°C (Holman)
        waterCp = 4174.0                    ! (Holman)
        waterSaturationTemperature = 100.0
        waterPr = 0.9                               ! Prandtl number of water
        waterk = 0.68                               ! Thermal conductivity of water
        waterViscosity = 0.003                      ! Kinematic viscosity of water in m2/s

        coolingwaterVolumetricFlowRate = 1.0 / 3600.0       ! m3/hr to m3/s
        coolingwaterDropSpeed = 1.0                         ! m/s
        coolingwaterDropDiameter = 1.0 / 1000.0     ! mm to m
        coolingwaterWaterTemperature = ambientTemperature
        coolingwaterBlockArea = 2 * Pi * blockL ^ 2 * (1 - COS(potPhi_Radians)) + Pi * (blockrt + blockR) *
SQRT((blockZ - blockzt) ^ 2 + (blockR - blockrt) ^ 2)
        coolingwaterVolumetricFlux = coolingwaterVolumetricFlowRate / coolingwaterBlockArea

        coolingwaterRe = coolingwaterVolumetricFlux * coolingwaterDropDiameter / waterViscosity
        coolingwaterNu = 2.512 * coolingwaterRe ^ 0.76 * waterPr ^ 0.56
        coolingwaterTIncip = 13.43 * coolingwaterRe ^ 0.167 * waterPr ^ 0.123 * (waterk /
coolingwaterDropDiameter) ^ 0.22 + coolingwaterWaterTemperature
        coolingwaterTcrit = 18 * ((waterVapourDensity * waterLatentHeat * coolingwaterVolumetricFlux) *
(waterSurfaceTension / (waterDensity * coolingwaterVolumetricFlux^ 2 * coolingwaterDropDiameter)) ^ 0.198)
^ (1 / 5.55) + coolingwaterWaterTemperature
        coolingwaterTmin = 204.895 * coolingwaterVolumetricFlux ^ 0.066 * coolingwaterDropSpeed ^ 0.138 *
coolingwaterDropDiameter ^ (-0.035) + coolingwaterWaterTemperature
        coolingwaterTdfb = 280.762 * coolingwaterVolumetricFlux ^ 0.087 * coolingwaterDropSpeed ^ 0.11 *
coolingwaterDropDiameter ^ (-0.035) + coolingwaterWaterTemperature
        coolingwaterHFdfb = 6100300.0 * coolingwaterVolumetricFlux ^ 0.588 * coolingwaterDropSpeed ^ 0.244
        coolingwaterHFmin = 3324400.0 * coolingwaterVolumetricFlux ^ 0.544 * coolingwaterDropSpeed ^ 0.324
        coolingwaterHFcrit = waterVapourDensity * waterLatentHeat * coolingwaterVolumetricFlux * 122.4 * (1
+ 0.0118 * (waterDensity / waterVapourDensity) ^ 0.25 * (waterDensity * waterCp *
(waterSaturationTemperature - coolingwaterWaterTemperature) / (waterVapourDensity * waterLatentHeat ))) *
(waterSurfaceTension / (waterDensity * coolingwaterVolumetricFlux ^ 2 * coolingwaterDropDiameter )) ^
0.0198

        coolingwaterN2 = (coolingwaterHFdfb - coolingwaterHFmin) / (coolingwaterTdfb - coolingwaterTmin) ^
2
        coolingwaterEN1 = -2 * coolingwaterN2 * coolingwaterTmin
        coolingwaterN0 = coolingwaterHFmin - coolingwaterEN1 * coolingwaterTmin - coolingwaterN2 *
coolingwaterTmin ^ 2

        coolingwaterh =  IF      temperature > coolingwaterTdfb THEN
                                        (63.25 * (temperature - coolingwaterWaterTemperature) ^ 1.691 *
coolingwaterVolumetricFlux ^ 0.264 / coolingwaterDropDiameter ^ 0.062) / (temperature -
coolingwaterWaterTemperature)
                                ELSE IF  temperature = coolingwaterTdfb THEN
                                        coolingwaterHFdfb / (temperature - coolingwaterWaterTemperature)
                                ELSE IF (temperature < coolingwaterTdfb) AND (temperature >
coolingwaterTmin) THEN
                                        ((coolingwaterN0 + coolingwaterEN1 * (temperature -
coolingwaterWaterTemperature) + coolingwaterN2 * (temperature - coolingwaterWaterTemperature) ^ 2)) /
(temperature - coolingwaterWaterTemperature)
                                ELSE IF temperature = coolingwaterTmin THEN
                                        coolingwaterHFmin / (temperature - coolingwaterWaterTemperature)
                                ELSE IF (temperature < coolingwaterTmin) AND (temperature >
coolingwaterTcrit) THEN
                                        (coolingwaterHFcrit - (coolingwaterHFcrit - coolingwaterHFmin) /
(coolingwaterTcrit - coolingwaterTmin) ^ 3 * (coolingwaterTcrit ^ 3 - 3 * coolingwaterTcrit ^ 2 *
coolingwaterTmin + 6 * coolingwaterTcrit * coolingwaterTmin * (temperature - coolingwaterWaterTemperature)
- 3 * (coolingwaterTcrit + coolingwaterTmin) * (temperature - coolingwaterWaterTemperature) ^ 2 + 2 *
(temperature - coolingwaterWaterTemperature) ^ 3)) / (temperature - coolingwaterWaterTemperature)
                                ELSE IF temperature = coolingwaterTcrit THEN
                                        coolingwaterHFcrit / (temperature - coolingwaterWaterTemperature)
                                ELSE IF (temperature < coolingwaterTcrit) AND (temperature >
coolingwaterTIncip) THEN
                                        (0.0000187 * (temperature - coolingwaterWaterTemperature) ^ 5.55)
/ (temperature - coolingwaterWaterTemperature)
                                ELSE    coolingwaterNu * waterk / coolingwaterDropDiameter

        ! Air cooling.
        solidsurfaceEmissivity = 0.8
        coolingairTf = 0.5 * (temperature + ambientTemperature)
        coolingairkf = 0.00005581 * coolingairTf + 0.02694
        coolingairb = 2 / (ambientTemperature + 273.15 + temperature + 273.15)
        coolingairv = 0.00000000006054 * coolingairTf ^ 2 + 0.000000102 * coolingairTf + 0.0000121
```

```
        coolingairPr = -9.11E-17 * coolingairTf ^ 5 + 5.007E-13 * coolingairTf ^ 4 - 0.000000001021 *
coolingairTf ^ 3 + 0.000000916 * coolingairTf ^ 2 - 0.0003139 * coolingairTf + 0.7153
        coolingairhRadiation = solidsurfaceEmissivity * 5.669 * (((temperature + 273.15) / 100) ^ 4 -
((ambientTemperature + 273) / 100) ^ 4) / (temperature - ambientTemperature) !equation 8-126, p471, Holman

        coolingairHorizontal_L = 0.25 * blockR * 2
        coolingairHorizontal_Gr = g * coolingairb * (temperature - ambientTemperature) *
coolingairHorizontal_L ^ 3 / coolingairv ^ 2
        coolingairHorizontal_C = IF coolingairHorizontal_Gr <= 8.0E6 THEN 0.54 ELSE 0.15
        coolingairHorizontal_m = IF coolingairHorizontal_Gr <= 8.0E6 THEN 1.0/4.0 ELSE 1.0/3.0
        coolingairHorizontal_hConvection = (coolingairkf / coolingairHorizontal_L) * coolingairHorizontal_C
*( (MAX(coolingairHorizontal_Gr, 0.0) * coolingairPr) ^ coolingairHorizontal_m)
        coolingairHorizontal_h = coolingairhRadiation + coolingairHorizontal_hConvection

        coolingairVertical_L = Cos(potTheta_Radians) / blockZ
        coolingairVertical_Gr = (g * coolingairb * (temperature - ambientTemperature) *
coolingairVertical_L ^ 3 / coolingairv ^ 2) * (Cos(potTheta_Radians)) ^ 2
        coolingairVertical_C = IF coolingairVertical_Gr <= 1.0E9 THEN 0.59 ELSE 0.1
        coolingairVertical_m = IF coolingairVertical_Gr <= 1.0E9 THEN 1.0/4.0 ELSE 1.0/3.0
        coolingairVertical_hConvection = coolingairkf / coolingairVertical_L * coolingairVertical_C *
(coolingairVertical_Gr * coolingairPr) ^ coolingairVertical_m
        coolingairVertical_h = coolingairhRadiation + coolingairVertical_hConvection

        ! Boundary Condition Schedule
        endTimeCoolingStep00 = 3600*18                                        ! Block in pot.

        endTimeCoolingStep01 =  endTimeCoolingStep00 + 3600
        endTimeCoolingStep02 =  endTimeCoolingStep01 + 3600
        endTimeCoolingStep03 =  endTimeCoolingStep02 + 3600
        endTimeCoolingStep04 =  endTimeCoolingStep03 + 3600
        endTimeCoolingStep05 =  endTimeCoolingStep04 + 3600
        endTimeCoolingStep06 =  endTimeCoolingStep05 + 3600
        endTimeCoolingStep07 =  endTimeCoolingStep06 + 3600
        endTimeCoolingStep08 =  endTimeCoolingStep07 + 3600
        endTimeCoolingStep09 =  endTimeCoolingStep08 + 3600
        endTimeCoolingStep10 =  endTimeCoolingStep09 + 3600

        ! Boundary Conditions
        horizontalAirCooling = URAMP(t - 0.0, t - 5.0) * coolingairHorizontal_h * (temperature -
ambientTemperature)
        verticalAirCooling = coolingairVertical_h * (temperature - ambientTemperature)
        verticalWaterCooling = coolingwaterh * (temperature - coolingwaterWaterTemperature )

        blockVerticalCooling =    IF      t <=endTimeCoolingStep00 THEN
                                                  0.0
{                                         ELSE IF  t <=endTimeCoolingStep01 THEN
                                                  verticalAirCooling
                                          ELSE IF  t <=endTimeCoolingStep02 THEN
                                                  verticalWaterCooling
                                          ELSE IF  t <=endTimeCoolingStep03 THEN
                                                  verticalAirCooling
                                          ELSE IF  t <=endTimeCoolingStep04 THEN
                                                  verticalWaterCooling
                                          ELSE IF  t <=endTimeCoolingStep05 THEN
                                                  verticalAirCooling
                                          ELSE IF  t <=endTimeCoolingStep06 THEN
                                                  verticalWaterCooling
                                          ELSE IF  t <=endTimeCoolingStep07 THEN
                                                  verticalAirCooling
                                          ELSE IF  t <=endTimeCoolingStep08 THEN
                                                  verticalWaterCooling
                                          ELSE IF  t <=endTimeCoolingStep09 THEN
                                                  verticalAirCooling
                                          ELSE IF  t <=endTimeCoolingStep10 THEN
                                                  verticalWaterCooling}
                                          ELSE 0.0

        blockHorizontalCooling =  IF      t <=endTimeCoolingStep00 THEN
                                                  horizontalAirCooling
{                                         ELSE IF  t <=endTimeCoolingStep01 THEN
                                                  horizontalContactCooling
                                          ELSE IF  t <=endTimeCoolingStep02 THEN
                                                  horizontalContactCooling
                                          ELSE IF  t <=endTimeCoolingStep03 THEN
                                                  horizontalContactCooling
                                          ELSE IF  t <=endTimeCoolingStep04 THEN
                                                  horizontalContactCooling
                                          ELSE IF  t <=endTimeCoolingStep05 THEN
                                                  horizontalContactCooling
                                          ELSE IF  t <=endTimeCoolingStep06 THEN
                                                  horizontalContactCooling
                                          ELSE IF  t <=endTimeCoolingStep07 THEN
                                                  horizontalContactCooling
                                          ELSE IF  t <=endTimeCoolingStep08 THEN
                                                  horizontalContactCooling
                                          ELSE IF  t <=endTimeCoolingStep09 THEN
                                                  horizontalContactCooling
                                          ELSE IF  t <=endTimeCoolingStep10 THEN
```

```
                                                        horizontalContactCooling}
                                          ELSE    0.0

        potVerticalCooling =              IF      t <=endTimeCoolingStep00 THEN
                                                  verticalAirCooling
                                          ELSE    verticalAirCooling

        ! Other Calculations
        liquidFraction = min(max((temperature -slagSolidusTemperature_0)/( slagLiquidusTemperature -
slagSolidusTemperature_0),0),1)

INITIAL VALUES
        temperature = initialTemperature
        !liquidFraction = initialLiquidFraction

EQUATIONS
        temperature:     slagDensity*Cp*dt(temperature) - div(k*grad(temperature))  = 0.0

BOUNDARIES

        REGION 1 'Block'
                initialTemperature = slagTemperature_AtTap
                k = slagk
                Cp = slagCp
                                                        START           (0.0, -blockL)
                NOBC(temperature)                       ARC(CENTER=0.0, 0.0) TO (blockrt, blockzt)
                NOBC(temperature)                       LINE    TO      (blockR, blockZ - blockL)
                NATURAL(temperature)=blockHorizontalCooling     LINE    TO      (0.0, blockZ - blockL)
                NOBC(temperature)                       LINE    TO      CLOSE

        {REGION 1 'Block'
                k = slagk
                Cp = slagCp
                                                        START           (0.0, -blockL)
                NATURAL(temperature)=blockVerticalCooling  ARC(CENTER=0.0, 0.0) TO (blockrt, blockzt)
                NATURAL(temperature)=blockVerticalCooling  LINE    TO      (blockR, blockZ - blockL)
                NATURAL(temperature)=blockHorizontalCooling     LINE    TO      (0.0, blockZ - blockL)
                NOBC(temperature)                       LINE    TO      CLOSE}

        REGION 2 'Pot'
                initialTemperature = potTemperature_BeforeTap
                k = steelk
                Cp = steelCp
                                                        START           (0.0, -blockL)
                NOBC(temperature)                       ARC(CENTER=0.0, 0.0) TO (blockrt, blockzt)
                NOBC(temperature)                       LINE    TO      (blockR, blockZ - blockL)
                NATURAL(temperature)=horizontalAirCooling  LINE    TO      (blockR + potWallThickness /
cos(potTheta_Radians), blockZ - blockL)
                NATURAL(temperature)=verticalAirCooling         LINE    TO      (blockrt +
potWallThickness * sin(potPhi_Radians), blockzt - potWallThickness * cos(potPhi_Radians))
                NATURAL(temperature)=verticalAirCooling         ARC(CENTER=0.0, 0.0) TO (0.0, -blockL -
potWallThickness)
                NOBC(temperature)                       LINE    TO      CLOSE

        FEATURE 'Block Vertical Outer Surface'
                START           (0.0, -blockL)
                ARC(CENTER=0.0, 0.0) TO (blockrt, blockzt)
                LINE    TO      (blockR, blockZ - blockL)

        FEATURE 'Pot Vertical Outer Surface'
                START           (0.0, -blockL - potWallThickness)
                ARC(CENTER=0.0, 0.0) TO (blockrt + potWallThickness * sin(potPhi_Radians), blockzt -
potWallThickness * cos(potPhi_Radians))
                LINE    TO      (blockR + potWallThickness / cos(potTheta_Radians), blockZ - blockL)

        FEATURE 'Pot Horizontal edge'
                START           (blockR + potWallThickness / cos(potTheta_Radians), blockZ - blockL)
                LINE    TO      (blockR, blockZ - blockL)

        FEATURE 'Block Horizontal Outer Surface'
                START           (blockR, blockZ - blockL)
                LINE    TO      (0.0, blockZ - blockL)

        FEATURE 'Block Centre Line'
                START           (0.0, -blockL)
                LINE TO  (0.0, -blockL + blockZ)

TIME
        0.0 BY 1E-1 TO endTimeCoolingStep00
        {endTimeCoolingStep00 BY 1E-1 TO endTimeCoolingStep10}

MONITORS
        FOR t= 0, 0.1, 0.5, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 25, 30, 45, 60, 300, 600 BY 600 TO 3600
BY 3600 TO ENDTIME
        CONTOUR(temperature) RANGE=(0, 1700)
!       CONTOUR(temperature) PAINTED RANGE=(0, 1700)
!       CONTOUR(liquidFraction2) PAINTED RANGE=(0.0, 1.0)
!       CONTOUR(liquidFraction) PAINTED fixed RANGE=(0.0, 1.0)
```

```
!       ELEVATION(temperature) ON 'Block Vertical Outer Surface'
!       ELEVATION(verticalAirCooling) ON 'Block Vertical Outer Surface'
!       ELEVATION(temperature) ON 'Block Horizontal Outer Surface'
!       ELEVATION(horizontalAirCooling) ON 'Block Horizontal Outer Surface'
!       ELEVATION(slagk) ON 'block horizontal outer surface'
!       ELEVATION(steelCp, steelk, temperature) ON 'Block Vertical Outer Surface'

PLOTS
!       FOR t= 0 BY 24*3600 TO ENDTIME

!       CONTOUR(temperature) RANGE=(0, 1700)
!       CONTOUR(temperature) PAINTED RANGE=(0, 1700)
!       CONTOUR(max(min(liquidFraction, 1.0), 0.0)) PAINTED RANGE=(0.0, 1.0)
!       ELEVATION(temperature) ON 'Block Vertical Outer Surface' EXPORT FORMAT "#d,#1"
!       ELEVATION(verticalAirCooling) ON 'Block Vertical Outer Surface'
!       ELEVATION(temperature) ON 'Block Horizontal Outer Surface'
!       ELEVATION(horizontalAirCooling) ON 'Block Horizontal Outer Surface'
!       TABLE(temperature) EXPORT FORMAT "#t,#x,#y,#1"

!       PLANT BLOCK
        FOR t = 0, 0.25, 0.5, 1, 2, 4, 8, 16, 32, 64, 100, 200, 400, 800, 1600, 3200
!       ELEVATION(liquidFraction) ON 'Block Centre Line' EXPORT FORMAT "#y, #1"
!       ELEVATION(verticalAirCooling) ON 'Block Vertical Outer Surface'      EXPORT FORMAT "#d, #1"
!       ELEVATION(horizontalAirCooling) ON 'Block Horizontal Outer Surface' EXPORT FORMAT "#d, #1"
        ELEVATION(coolingairHorizontal_h) ON 'Block Horizontal Outer Surface' EXPORT FORMAT "#d, #1"
        ELEVATION(steelk) ON 'Block Vertical Outer Surface' EXPORT FORMAT "#d, #1"

        SUMMARY
                REPORT potMassCapacity
                REPORT tapMass
                REPORT potR
                REPORT potTheta_Degrees
                REPORT potTheta_Radians
                REPORT potWallThickness

                REPORT potVolume
                REPORT blockVolume
                REPORT potPhi_Degrees
                REPORT potPhi_Radians
                REPORT blockL
                REPORT blockrt
                REPORT blockzt
                REPORT blockVolume_SpericalCap
                REPORT blockVolume_ConicalSection
                REPORT blockR
                REPORT blockZ
                REPORT slagSolidusTemperature_0
                REPORT slagLiquidusTemperature

        FOR t = ENDTIME
!       TRANSFER(temperature, liquidFraction) FILE="transfer.dat"
        TRANSFER(temperature) FILE="transfer.dat"

HISTORIES
!       HISTORY(SURF_INTEGRAL(blockVerticalCooling, 'Block Vertical Outer Surface') +
SURF_INTEGRAL(potVerticalCooling, 'Pot Vertical Outer Surface') ) AS 'HeatLossV' EXPORT FORMAT "#t#r,#i"
!       HISTORY(SURF_INTEGRAL(blockHorizontalCooling, 'Block Horizontal Outer Surface')  +
SURF_INTEGRAL(horizontalAirCooling,'Pot Horizontal edge')) AS 'HeatLossH' EXPORT FORMAT "#t#r,#i"
!       HISTORY(VOL_INTEGRAL(slagEnthalpy*slagDensity, 1)+VOL_INTEGRAL(steelCp*slagDensity*(temperature-
25), 2))  AS 'TotalEnergy' EXPORT FORMAT "#t#r,#i"
        {Tap 37 & 38 C10 thermocouple positions}
!       HISTORY(temperature) AT (0,0.13)  AT(0,-0.020) AT(0,-0.120)  AS 'ThermocouplePositions' EXPORT
FORMAT "#t#r,#i"

!       TAP 42 &/ 38
!       HISTORY(SURF_INTEGRAL(verticalAirCooling, 'Block vertical outer surface')) AS 'Heatlosses from Pot
surface' EXPORT FORMAT "#t,#1"
!       HISTORY(temperature) AT (0.0, -blockL - potWallThickness) AT(blockrt + potWallThickness *
sin(potPhi_Radians), blockzt - potWallThickness * cos(potPhi_Radians)) AT(blockR + potWallThickness /
cos(potTheta_Radians), blockZ - blockL) AS 'Pot surface temperatures' EXPORT FORMAT "#t#r,#i"

 !      PLANT TAPS
!       HISTORY(temperature)  AT (0.0, blockZ - blockL) AT (blockR, blockZ - blockL) AT (blockrt, blockzt)
AT (0.0, -blockL)  AS  'Block surface temperatures' EXPORT FORMAT "#t #r, #i"

END
```

File 2: slag block

```
! ###################################################################################################
!  UP Slag Block Cooling 3b.pde
!
!       Client:          Department of Material Science and Metallurgical Engineering
!                        University of Pretoria
!       Date:            2007-02-13
!       Drawing: ???
!
!       Developer:       Johan Zietsman
!
!       Model Type:      Transient
!       Model Purpose:   The model is part of Hanlie Kotze's Ph.D. study.  Chris Pistorius requested Ex
Mente to assist with the development of a
!                        model based on Hanlie's formulation. Hanlie would then use this model to
!                        generate results required for the completion of her Ph.D.
!                        The purpose of the model is to describe the cooling and solidification
!                        of a high-TiO2 slag block as it moves through the various cooling stages
!                        and cooling conditions on the plant.
!                        The model must take into account the following factors:
!                        - tap temperature,
!                        - tap composition,
!                        - cooling conditions.
!                        The model must also be able to describe a small pilot-plant configuration
!                        and the full-scale industrial configuration.
!
!       Units:           Temperature              [°C]
!                        Distance         [m]
!                        Time                     [s]
!                        Fraction         []
!                        Chemical Composition     [%]
!                        Density                  [kg/m3]
!                        Mass                     [kg]
!                        Heat Capacity            [J/(kg.°C)]
!                        Thermal Conductivity     [W/(m.°C)]
!
! ###################################################################################################

TITLE
        'UP Slag Block Cooling 5b'

COORDINATES
        YCYLINDER

SELECT
        ERRLIM = 1E-4
        SMOOTHINIT
        GRAY

VARIABLES
        temperature (1700)

DEFINITIONS
        ! ###########################################################################################
        ! Constants
        ! ###########################################################################################

        g = 9.81          ! Gravitational acceleration      [m/s2]
        sigma = 5.669E-8  ! Stefan-Boltzmann constant       [W/(m2.K4)]
        slagDensity = 3800.0      ! Slag density                    [kg/m3]

        ! ###########################################################################################
        ! Model Inputs
        ! ###########################################################################################

!       %%% Tap 37
{       slagFeOContent_Analysed = 13.25
        slagTemperature_AtTap = 1669.0
        tapmass = 1002.00}

!       %%% Tap 38
{       slagFeOContent_Analysed = 13.92
        slagTemperature_AtTap = 1668.0
        tapmass = 1365.00}

!       %%%% TAP42 C9 WATERcooling
{       slagFeOContent_Analysed = 11.99
        slagTemperature_AtTap = 1709.0
        tapMass = 1017.0}

!       %%% Tap 64 C9 AIRcooling
{       slagFeOContent_Analysed = 11.46
        slagTemperature_AtTap = 1683.0
        tapMass = 1420.0}
```

```
!       Big tap
        slagFeOContent_Analysed = 10.10
        slagTemperature_AtTap = 1678.0
        tapMass = 18200.0

        potTemperature_BeforeTap = 25.0
        slagFeOContent = slagFeOContent_Analysed
        slagLiquidusTemperature = (0.2351*slagFeOContent^2 - 11.24*slagFeOContent + 1664.1)
        slagSolidusTemperature_0 = (0.0364*slagFeOContent^2 - 4.845*slagFeOContent + 1502.7)

        ! Domain Dimension Inputs (1.5 t)
{       potMassCapacity = 1500.0
        potR = 0.523
        potTheta_Degrees = 15.0
        potTheta_Radians = (potTheta_Degrees / 360.0) * 2.0 * PI
        potWallThickness = 0.04}

        ! Domain Dimension Inputs (20 t)
        potMassCapacity = 20000.0
        potR = 1.175
        potTheta_Degrees = 15.0
        potTheta_Radians = (potTheta_Degrees / 360.0) * 2.0 * PI
        potWallThickness = 0.11

        ! ###########################################################################################
        ! Domain Dimensions
        ! ###########################################################################################

        ! Domain Dimension Calculations
        potVolume = potMassCapacity / slagDensity
        blockVolume = tapMass / slagDensity
        potPhi_Degrees = 90.0 - potTheta_Degrees
        potPhi_Radians = (potPhi_Degrees / 360.0) * 2.0 * PI
        blockL  =  ((potVolume  -  PI  *  potR  ^  3.0  /  3.0  /  tan(potTheta_Radians))  /  (-PI  *
(cos(potTheta_Radians)) ^ 3.0 / 3.0 / tan(potTheta_Radians) + PI / 3.0 * (2.0 + cos(potPhi_Radians)) * (1.0
- cos(potPhi_Radians)) ^ 2.0)) ^ (1.0 / 3.0)
        blockrt = blockL*cos(potTheta_Radians)
        blockzt = blockL*(1.0 - sin(potTheta_Radians)) - blockL
        blockVolume_SpericalCap  =  (PI/3.0)  *  blockL^3.0  *  (2.0  +  cos(potPhi_Radians))  *  (1.0  -
cos(potPhi_Radians))^2.0
        blockVolume_ConicalSection = (PI/3.0) / tan(potTheta_Radians) * (potR^3.0 - blockrt^3.0)
        blockR    =    ((3.0*tan(potTheta_Radians)/PI)*(blockVolume    -    blockVolume_SpericalCap)    +
blockrt^3.0)^(1.0/3.0)
        blockZ = (blockR - blockrt)/tan(potTheta_Radians) + blockzt + blockL

        ! ###########################################################################################
        ! Conditions
        ! ###########################################################################################

        ambientTemperature = 25.0

        ! ###########################################################################################
        ! Physical Properties
        ! ###########################################################################################

        k = 0.0          ! Thermal conductivity
        Cp = 0.0 ! Heat capacity

        ! Slag
        slagk = (max((0.00175*temperature+0.3),0.3))*1

        slagCp_Solid = - 0.0314*slagFeOContent^2 - 0.4042*slagFeOContent^2 + 908.51
        slagCp_Liquid = 0.0561*SlagFeOContent^2 - 3.3668*slagFeOContent + 1044.3
        slagEnthalpy_At25 = - 139.51*slagFeOContent^2 - 1086.1*slagFeOContent + 515805

        slagCp=  if (temperature <= slagSolidusTemperature_0) THEN slagCp_Solid
                         ELSE    IF (temperature >= slagLiquidusTemperature) THEN slagCp_Liquid
                         ELSE   (slagEnthalpy_At25   +   slagCp_Liquid*(slagLiquidusTemperature   -   25)   -
slagCp_Solid*(slagSolidusTemperature_0 - 25)) / (slagLiquidusTemperature-slagSolidusTemperature_0)

        ! Steel
!       steelk = 54.0            ! Carbon Steel 0.5% (Holman)
!       steelCp = 465.0  ! Carbon Steel 0.5% (Holman)

        ! ###########################################################################################
        ! Initial Conditions
        ! ###########################################################################################

        !initialTemperature = 25.0
        TRANSFER("transfer_1.dat", initialTemperature)

        ! ###########################################################################################
        ! Boundary Conditions
        ! ###########################################################################################

        ! Water cooling.
        waterVapourDensity = 0.5863              ! At 107°C (Holman)
```

```
        waterLatentHeat = 2256000.0                 ! Latent heat of water to gas reaction, J/kg (internet
data)
        waterSurfaceTension = 0.0588                ! [N/m] (internet data)
        waterDensity = 994.9                        ! At 32°C (Holman)
        waterCp = 4174.0                    ! (Holman)
        waterSaturationTemperature = 100.0
        waterPr = 0.9                               ! Prandtl number of water
        waterk = 0.68                               ! Thermal conductivity of water
        waterViscosity = 0.003                      ! Kinematic viscosity of water in m2/s

!       &&& TUNING PARAMETERS
        coolingwaterVolumetricFlowRate = 1 / 3600.0         ! m3/hr to m3/s
        coolingwaterDropSpeed = 1                   ! m/s
        coolingwaterDropDiameter = 1 / 1000.0               ! mm to m
!       &&&&&&&&&&&&&&&&

        coolingwaterWaterTemperature = ambientTemperature
        coolingwaterBlockArea = 2 * Pi * blockL ^ 2 * (1 - COS(potPhi_Radians)) + Pi * (blockrt + blockR) *
SQRT((blockZ - blockzt) ^ 2 + (blockR - blockrt) ^ 2)
        coolingwaterVolumetricFlux = coolingwaterVolumetricFlowRate / coolingwaterBlockArea

        coolingwaterRe = coolingwaterVolumetricFlux * coolingwaterDropDiameter / waterViscosity
        coolingwaterNu = 2.512 * coolingwaterRe ^ 0.76 * waterPr ^ 0.56
        coolingwaterTIncip  =  13.43  *  coolingwaterRe  ^  0.167  *  waterPr  ^  0.123  *  (waterk  /
coolingwaterDropDiameter) ^ 0.22 + coolingwaterWaterTemperature
        coolingwaterTcrit = 18 * ((waterVapourDensity * waterLatentHeat * coolingwaterVolumetricFlux) *
(waterSurfaceTension / (waterDensity * coolingwaterVolumetricFlux^ 2 * coolingwaterDropDiameter)) ^ 0.198)
^ (1 / 5.55) + coolingwaterWaterTemperature
        coolingwaterTmin = 204.895 * coolingwaterVolumetricFlux ^ 0.066 * coolingwaterDropSpeed ^ 0.138 *
coolingwaterDropDiameter ^ (-0.035) + coolingwaterWaterTemperature
        coolingwaterTdfb = 280.762 * coolingwaterVolumetricFlux ^ 0.087 * coolingwaterDropSpeed ^ 0.11 *
coolingwaterDropDiameter ^ (-0.035) + coolingwaterWaterTemperature
        coolingwaterHFdfb = 6100300.0 * coolingwaterVolumetricFlux ^ 0.588 * coolingwaterDropSpeed ^ 0.244
        coolingwaterHFmin = 3324400.0 * coolingwaterVolumetricFlux ^ 0.544 * coolingwaterDropSpeed ^ 0.324
        coolingwaterHFcrit = waterVapourDensity * waterLatentHeat * coolingwaterVolumetricFlux * 122.4 * (1
+   0.0118   *   (waterDensity   /   waterVapourDensity)   ^   0.25   *   (waterDensity   *   waterCp   *
(waterSaturationTemperature - coolingwaterWaterTemperature) / (waterVapourDensity * waterLatentHeat ))) *
(waterSurfaceTension / (waterDensity * coolingwaterVolumetricFlux ^ 2 * coolingwaterDropDiameter )) ^
0.0198

        coolingwaterN2 = (coolingwaterHFdfb - coolingwaterHFmin) / (coolingwaterTdfb - coolingwaterTmin) ^
2
        coolingwaterEN1 = -2 * coolingwaterN2 * coolingwaterTmin
        coolingwaterN0  =  coolingwaterHFmin  -  coolingwaterEN1  *  coolingwaterTmin  -  coolingwaterN2  *
coolingwaterTmin ^ 2

        coolingwaterh =         IF      temperature > coolingwaterTdfb THEN
                                        (63.25 * (temperature - coolingwaterWaterTemperature) ^ 1.691 *
coolingwaterVolumetricFlux   ^   0.264   /   coolingwaterDropDiameter   ^   0.062)   /   (temperature   -
coolingwaterWaterTemperature)
                                ELSE IF  temperature = coolingwaterTdfb THEN
                                        coolingwaterHFdfb / (temperature - coolingwaterWaterTemperature)
                                ELSE   IF   (temperature   <   coolingwaterTdfb)   AND   (temperature   >
coolingwaterTmin) THEN
                                        ((coolingwaterN0    +    coolingwaterEN1    *    (temperature    -
coolingwaterWaterTemperature)  +  coolingwaterN2  *  (temperature  -  coolingwaterWaterTemperature)  ^  2))  /
(temperature - coolingwaterWaterTemperature)
                                ELSE IF temperature = coolingwaterTmin THEN
                                        coolingwaterHFmin / (temperature - coolingwaterWaterTemperature)
                                ELSE   IF   (temperature   <   coolingwaterTmin)   AND   (temperature   >
coolingwaterTcrit) THEN
                                        (coolingwaterHFcrit - (coolingwaterHFcrit - coolingwaterHFmin) /
(coolingwaterTcrit  -  coolingwaterTmin)  ^  3  *  (coolingwaterTcrit  ^  3  -  3  *  coolingwaterTcrit  ^  2  *
coolingwaterTmin + 6 * coolingwaterTcrit * coolingwaterTmin * (temperature - coolingwaterWaterTemperature)
- 3 * (coolingwaterTcrit + coolingwaterTmin) * (temperature - coolingwaterWaterTemperature) ^ 2 + 2 *
(temperature - coolingwaterWaterTemperature) ^ 3)) / (temperature - coolingwaterWaterTemperature)
                                ELSE IF temperature = coolingwaterTcrit THEN
                                        coolingwaterHFcrit / (temperature - coolingwaterWaterTemperature)
                                ELSE   IF   (temperature   <   coolingwaterTcrit)   AND   (temperature   >
coolingwaterTIncip) THEN
                                        (0.0000187 * (temperature - coolingwaterWaterTemperature) ^ 5.55)
/ (temperature - coolingwaterWaterTemperature)
                                ELSE    coolingwaterNu * waterk / coolingwaterDropDiameter

        ! Air cooling.
        solidsurfaceEmissivity = 0.8
        coolingairTf = 0.5 * (temperature + ambientTemperature)
        coolingairkf = 0.00005581 * coolingairTf + 0.02694
        coolingairb = 2 / (ambientTemperature + 273.15 + temperature + 273.15)
        coolingairv = 0.00000000006054 * coolingairTf ^ 2 + 0.000000102 * coolingairTf + 0.0000121
        coolingairPr = -9.11E-17 * coolingairTf ^ 5 + 5.007E-13 * coolingairTf ^ 4 - 0.000000001021 *
coolingairTf ^ 3 + 0.000000916 * coolingairTf ^ 2 - 0.0003139 * coolingairTf + 0.7153
        coolingairhRadiation  =  solidsurfaceEmissivity  *  5.669  *  (((temperature  +  273)  /  100)  ^  4  -
((ambientTemperature + 273) / 100) ^ 4) / (temperature - ambientTemperature) !equation 8-126, p471, Holman

        coolingairHorizontal_L = 0.25 * blockR * 2
        coolingairHorizontal_Gr   =   g   *   coolingairb   *   (temperature   -   ambientTemperature)   *
coolingairHorizontal_L ^ 3 / coolingairv ^ 2
```

```
        coolingairHorizontal_C = IF coolingairHorizontal_Gr <= 8.0E6 THEN 0.54 ELSE 0.15
        coolingairHorizontal_m = IF coolingairHorizontal_Gr <= 8.0E6 THEN 1.0/4.0 ELSE 1.0/3.0
        coolingairHorizontal_hConvection = coolingairkf / coolingairHorizontal_L * coolingairHorizontal_C *
(coolingairHorizontal_Gr * coolingairPr) ^ coolingairHorizontal_m
        coolingairHorizontal_h = coolingairhRadiation + coolingairHorizontal_hConvection

        coolingairVertical_L = Cos(potTheta_Radians) / blockZ
        coolingairVertical_Gr = (g * coolingairb * (temperature - ambientTemperature) *
coolingairVertical_L ^ 3 / coolingairv ^ 2) * (Cos(potTheta_Radians)) ^ 2
        coolingairVertical_C = IF coolingairVertical_Gr <= 1.0E9 THEN 0.59 ELSE 0.1
        coolingairVertical_m = IF coolingairVertical_Gr <= 1.0E9 THEN 1.0/4.0 ELSE 1.0/3.0
        coolingairVertical_hConvection = coolingairkf / coolingairVertical_L * coolingairVertical_C *
(coolingairVertical_Gr * coolingairPr) ^ coolingairVertical_m
        coolingairVertical_h = coolingairhRadiation + coolingairVertical_hConvection

        ! Boundary Condition Schedule

!       PLANT BLOCKS LANE 1
{       endTimeCoolingStep00 = 3600*17.75                           ! Block in pot.
        endTimeCoolingStep01 =  endTimeCoolingStep00 + 3600*0.25    ! Air cooling
        endTimeCoolingStep02 =  endTimeCoolingStep01 + 3600*6.6
        endTimeCoolingStep03 =  endTimeCoolingStep02 + 3600*0.1
        endTimeCoolingStep04 =  endTimeCoolingStep03 + 3600*75.7
        endTimeCoolingStep05 =  endTimeCoolingStep04 + 3600*0.1
        endTimeCoolingStep06 =  endTimeCoolingStep05 + 3600*0.1
        endTimeCoolingStep07 =  endTimeCoolingStep06 + 3600*58.8
        endTimeCoolingStep08 =  endTimeCoolingStep07 + 3600*0.3
        endTimeCoolingStep09 =  endTimeCoolingStep08 + 3600*7.2
        endTimeCoolingStep10 =  endTimeCoolingStep09 + 3600*0.5
        endTimeCoolingStep11 =  endTimeCoolingStep10 + 3600*145.2
        endTimeCoolingStep12 =  endTimeCoolingStep11 + 3600*0.12
        endTimeCoolingStep13 =  endTimeCoolingStep12 + 3600
        endTimeCoolingStep14 =  endTimeCoolingStep13 + 3600}

!       PLANT BLOCKS LANE 2
        endTimeCoolingStep00 = 3600*18                                  ! Block in pot.
        endTimeCoolingStep01 =  endTimeCoolingStep00 + 3600*24*15   ! Air cooling
!       endTimeCoolingStep02 = endTimeCoolingStep00 + 3600*24*15    ! Water cooling
!       endTimeCoolingStep03 = endTimeCoolingStep02 + 3600*1        ! Air cooling

 !      TAP 42 AIR & WATER COOLING
{       endTimeCoolingStep00 = 3600*17.75 ! Block in pot.
        endTimeCoolingStep01 =  endTimeCoolingStep00 + 3600*0.25    ! Air cooling
        endTimeCoolingStep02 =  endTimeCoolingStep01 + 3600*15      ! Water cooling
        endTimeCoolingStep03 =  endTimeCoolingStep02 + 3600*0.5
        endTimeCoolingStep04 =  endTimeCoolingStep03 + 3600*1.2
        endTimeCoolingStep05 =  endTimeCoolingStep04 + 3600*0.5
        endTimeCoolingStep06 =  endTimeCoolingStep05 + 3600*1.1
        endTimeCoolingStep07 =  endTimeCoolingStep06 + 3600*0.6
        endTimeCoolingStep08 =  endTimeCoolingStep07 + 3600*1.0
        endTimeCoolingStep09 =  endTimeCoolingStep08 + 3600*0.7
        endTimeCoolingStep10 =  endTimeCoolingStep09 + 3600*1.0
        endTimeCoolingStep11 =  endTimeCoolingStep10 + 3600*1.0
        endTimeCoolingStep12 =  endTimeCoolingStep11 + 3600*1.0
        endTimeCoolingStep13 =  endTimeCoolingStep12 + 3600*1.2
        endTimeCoolingStep14 =  endTimeCoolingStep13 + 3600*1.0}


        ! Boundary Conditions
        horizontalAirCooling = URAMP(t - 0.0, t - 1.0) * coolingairHorizontal_h * (temperature -
ambientTemperature)
        horizontalContactCooling = (coolingairkf / (0.050) + coolingairhRadiation) * (temperature -
ambientTemperature)
        verticalAirCooling = coolingairVertical_h * (temperature - ambientTemperature)
        verticalWaterCooling = coolingwaterh * (temperature - coolingwaterWaterTemperature )

        blockVerticalCooling =          IF      t <=endTimeCoolingStep00 THEN
                                                0.0
                                        ELSE IF t <=endTimeCoolingStep01 THEN
                                                verticalAirCooling
{                                       ELSE IF t <=endTimeCoolingStep02 THEN
                                                verticalWaterCooling
                                        ELSE IF t <=endTimeCoolingStep03 THEN
                                                verticalAirCooling
                                        ELSE IF t <=endTimeCoolingStep04 THEN
                                                verticalWaterCooling
                                        ELSE IF t <=endTimeCoolingStep05 THEN
                                                verticalAirCooling
                                        ELSE IF t <=endTimeCoolingStep06 THEN
                                                verticalWaterCooling
                                        ELSE IF t <=endTimeCoolingStep07 THEN
                                                verticalAirCooling
                                        ELSE IF t <=endTimeCoolingStep08 THEN
                                                verticalWaterCooling
                                        ELSE IF t <=endTimeCoolingStep09 THEN
                                                verticalAirCooling
                                        ELSE IF t <=endTimeCoolingStep10 THEN
                                                verticalWaterCooling
```

```
                                          ELSE IF  t <=endTimeCoolingStep11 THEN
                                                   verticalAirCooling
                                          ELSE IF  t <=endTimeCoolingStep12 THEN
                                                   verticalWaterCooling
                                          ELSE IF  t <=endTimeCoolingStep13 THEN
                                                   verticalAirCooling
                                          ELSE IF  t <=endTimeCoolingStep14 THEN
                                                   verticalWaterCooling}
                                          ELSE     0.0

        blockHorizontalCooling =  IF      t <=endTimeCoolingStep00 THEN
                                                   horizontalAirCooling
                                          ELSE IF  t <=endTimeCoolingStep01 THEN
                                                   horizontalContactCooling
{                                         ELSE IF  t <=endTimeCoolingStep02 THEN
                                                   horizontalContactCooling
                                          ELSE IF  t <=endTimeCoolingStep03 THEN
                                                   horizontalContactCooling
                                          ELSE IF  t <=endTimeCoolingStep04 THEN
                                                   horizontalContactCooling
                                          ELSE IF  t <=endTimeCoolingStep05 THEN
                                                   horizontalContactCooling
                                          ELSE IF  t <=endTimeCoolingStep06 THEN
                                                   horizontalContactCooling
                                          ELSE IF  t <=endTimeCoolingStep07 THEN
                                                   horizontalContactCooling
                                          ELSE IF  t <=endTimeCoolingStep08 THEN
                                                   horizontalContactCooling
                                          ELSE IF  t <=endTimeCoolingStep09 THEN
                                                   horizontalContactCooling
                                          ELSE IF  t <=endTimeCoolingStep10 THEN
                                                   horizontalContactCooling
                                          ELSE IF  t <=endTimeCoolingStep11 THEN
                                                   horizontalContactCooling
                                          ELSE IF  t <=endTimeCoolingStep12 THEN
                                                   horizontalContactCooling
                                          ELSE IF  t <=endTimeCoolingStep13 THEN
                                                   horizontalContactCooling
                                          ELSE IF  t <=endTimeCoolingStep14 THEN
                                                   horizontalContactCooling}
                                          ELSE     0.0

        potVerticalCooling =              IF       t <=endTimeCoolingStep00 THEN
                                                   verticalAirCooling
                                          ELSE     verticalAirCooling

        ! Other Calculations
        liquidFraction  =  min(max((temperature  -slagSolidusTemperature_0)/(  slagLiquidusTemperature  -
slagSolidusTemperature_0),0),1)
        liquidFraction2 = max(min(liquidFraction, 1.0), 0.0)


INITIAL VALUES
        temperature = initialTemperature

EQUATIONS
        temperature:     slagDensity*cp*dt(temperature) - div(k*grad(temperature))  = 0.0

BOUNDARIES

        {REGION 1 'Block'
                initialTemperature = slagTemperature_AtTap
                initialLiquidFraction = 1.0
                k = slagk
                Cp = slagCp
                                                        START            (0.0, -blockL)
                NOBC(temperature)                   ARC(CENTER=0.0, 0.0) TO (blockrt, blockzt)
                NOBC(temperature)                   LINE     TO      (blockR, blockZ - blockL)
                NATURAL(temperature)=blockHorizontalCooling   LINE     TO      (0.0, blockZ - blockL)
                NOBC(temperature)                   LINE     TO      CLOSE}

        REGION 1 'Block'
                k = slagk
                cp = slagCp
                                                        START            (0.0, -blockL)
                NATURAL(temperature)=blockVerticalCooling  ARC(CENTER=0.0, 0.0) TO (blockrt, blockzt)
                NATURAL(temperature)=blockVerticalCooling  LINE     TO      (blockR, blockZ - blockL)
                NATURAL(temperature)=blockHorizontalCooling   LINE     TO      (0.0, blockZ - blockL)
                NOBC(temperature)                   LINE     TO      CLOSE

        {REGION 2 'Pot'
                initialTemperature = potTemperature_BeforeTap
                initialLiquidFraction = 0.0
                k = steelk
                Cp = steelCp
                                                        START            (0.0, -blockL)
                NOBC(temperature)                   ARC(CENTER=0.0, 0.0) TO (blockrt, blockzt)
                NOBC(temperature)                   LINE     TO      (blockR, blockZ - blockL)
```

```
                NATURAL(temperature)=horizontalAirCooling  LINE    TO      (blockR  +  potWallThickness  /
cos(potTheta_Radians), blockZ - blockL)
                NATURAL(temperature)=verticalAirCooling            LINE    TO      (blockrt             +
potWallThickness * sin(potPhi_Radians), blockzt - potWallThickness * cos(potPhi_Radians))
                NATURAL(temperature)=verticalAirCooling            ARC(CENTER=0.0, 0.0) TO (0.0, -blockL -
potWallThickness)
                NOBC(temperature)                          LINE    TO      CLOSE}

        FEATURE 'Block Vertical Outer Surface'
                START           (0.0, -blockL)
                ARC(CENTER=0.0, 0.0) TO (blockrt, blockzt)
                LINE    TO      (blockR, blockZ - blockL)

        FEATURE 'Block Horizontal Outer Surface'
                START           (blockR, blockZ - blockL)
                LINE    TO      (0.0, blockZ - blockL)

        FEATURE 'Block Centre Line'
                START           (0.0, -blockL)
                LINE TO  (0.0, -blockL + blockZ)

        FEATURE 'Block Roundend Radius'
                START           (0.0, 0.0)
                LINE TO  (blockrt, blockzt)

TIME
        {0.0 BY 1E-1 TO endTimeCoolingStep00}
        endTimeCoolingStep00 BY 1E-1 TO endTimeCoolingStep01


MONITORS
        FOR t= 0, 0.1, 0.5, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 25, 30, 45, 60, 300, 600 BY 600 TO 3600
BY 3600 TO ENDTIME
        CONTOUR(temperature) RANGE=(0, 1700)
!       CONTOUR(temperature) PAINTED RANGE=(0, 1700)
!       CONTOUR(liquidFraction2) PAINTED RANGE=(0.0, 1.0)
!       ELEVATION(temperature) ON 'Block Vertical Outer Surface'
!       ELEVATION(verticalAirCooling) ON 'Block Vertical Outer Surface'
!       ELEVATION(temperature) ON 'Block Horizontal Outer Surface'
!       ELEVATION(horizontalAirCooling) ON 'Block Horizontal Outer Surface'

PLOTS
        FOR t=endTimeCoolingStep00 BY 3600*24 TO ENDTIME

!       CONTOUR(temperature) RANGE=(0, 1700)
!       CONTOUR(temperature) PAINTED RANGE=(0, 1700)
!       CONTOUR(max(min(liquidFraction, 1.0), 0.0)) PAINTED RANGE=(0.0, 1.0)
!       ELEVATION(temperature) ON 'Block Vertical Outer Surface'
!       ELEVATION(verticalAirCooling) ON 'Block Vertical Outer Surface'
!       ELEVATION(temperature) ON 'Block Vertical Outer Surface' EXPORT FORMAT "#d, #1"
        ELEVATION(horizontalAirCooling) ON 'Block Horizontal Outer Surface'

!       FOR t=0 BY 3600.00*24 TO ENDTIME
!       ELEVATION(temperature) ON 'Block Vertical Outer Surface' EXPORT FORMAT "#x,#y,#1"

!       PLANT BLOCK
!       FOR t = 0, 0.1, 0.5, 1, 2, 4, 6, 8, 10, 15, 20, 25, 30, 45, 60, 80, 100, 200, 300, 400, 500, 600 BY
600 TO ENDTIME
!       ELEVATION(liquidFraction) ON 'Block Centre Line' EXPORT FORMAT "#y, #1"
!       ELEVATION(temperature, coolingairvertical_h) ON 'Block Vertical Outer Surface' EXPORT FORMAT "#d,
#1, #2"
!       CONTOUR(temperature) RANGE=(0,1700) EXPORT FORMAT "#x, #y, #1"
!       ELEVATION(temperature) ON 'Block Roundend Radius' EXPORT FORMAT "#d, #1"

        SUMMARY
                REPORT potMassCapacity
                REPORT tapMass
                REPORT potR
                REPORT potTheta_Degrees
                REPORT potTheta_Radians
                REPORT potWallThickness

                REPORT potVolume
                REPORT blockVolume
                REPORT potPhi_Degrees
                REPORT potPhi_Radians
                REPORT blockL
                REPORT blockrt
                REPORT blockzt
                REPORT blockVolume_SpericalCap
                REPORT blockVolume_ConicalSection
                REPORT blockR
                REPORT blockZ
                REPORT slagSolidusTemperature_0
                REPORT slagLiquidusTemperature

        {FOR t = ENDTIME
        TRANSFER(temperature, liquidFraction) FILE="transfer.dat"}
```

```
HISTORIES
!       HISTORY(SURF_INTEGRAL(blockVerticalCooling, 'Block Vertical Outer Surface') / SURF_INTEGRAL(1, 'Block Vertical Outer Surface')) AS 'VerticalHeatLoss'
!       HISTORY(VOL_INTEGRAL(liquidFraction2, 1) / VOL_INTEGRAL(1, 1)) AS 'TotalLiquidFraction'

        {Tap 37 & 38 C10 thermocouple positions}
!       HISTORY(temperature) AT (0,0.13)  AT(0,-0.020) AT(0,-0.120)  AS 'ThermocouplePositions' EXPORT FORMAT "#t#r,#i"
!       HISTORY(temperature, coolingairVertical_h, coolingwaterh)  AT (0.0, blockZ - blockL) AT (blockR, blockZ - blockL) AT (blockrt, blockzt)  AT (0.0, -blockL)  AS  'Block surface temperatures' EXPORT FORMAT "#t #r, #i"

!       TAP 42
!       HISTORY(SURF_INTEGRAL(blockHorizontalCooling, 'Block Horizontal Outer Surface')) AS 'Heatloss to ground' EXPORT FORMAT "#t,#1"
!       HISTORY(SURF_INTEGRAL(blockVerticalCooling, 'Block Vertical Outer Surface')) AS 'Heatloss to suroundings' EXPORT FORMAT "#t,#1"

!       PLANT TAPS
!       HISTORY(temperature, coolingairvertical_h)  AT (0.0, blockZ - blockL) AT (blockR, blockZ - blockL) AT (blockrt, blockzt)  AT (0.0, -blockL)  AS  'Block surface temperatures' EXPORT FORMAT "#t #r, #i"


END
```

5.6 List of expressions used to calculate the heat transfer coefficient in spray cooling.[37,38]

5.7 Nomenclature

d_{32} = Sauter mean drop diameter, m

(the Sauter mean diameter is the drop diameter with a volume to surface area which is equal that of the entire water spray)

Nu =Nusselt number, dimensionless

Pr = Prandtl number, dimensionless

Re_{32} = Reynolds number based on the Sauter Mean Diameter[§§§], dimensionless

T = temperature, ˚C

Q'' =volumetric flux of the cooling water, m^3s^{-1}/m^2

q'' = heat flux, W/m^2

k = thermal conductivity, $W/m^2/^\circ C$

σ = surface tension of water, N/m

ρ = density, kg/m^3

υ = drop speed, m/s

ϑ = kinematic viscosity of water, m^2/s

h_{fg} = latent heat of vaporisation, J/kg

Subscripts

$incip$ = point of incipient nucleate boiling

$crit$ =point of critical heat flux

$\min$ = point of minimum heat flux (Leidenfrost point)

dfb = point of departure from film boiling

s = hot surface

w = water

v = vapour

$$\mathrm{Re}_{32} = \frac{Q^{"} d_{32}}{\vartheta}$$

$$Nu_f = 2.512\,\mathrm{Re}_{32}^{\;0.76}\,\mathrm{Pr}_w^{\;0.56}$$

$$T_{incip} = 13.43\,\mathrm{Re}_{32}^{\;0.167}\,\mathrm{Pr}_w^{\;0.123}\left(\frac{k_w}{d_{32}}\right)^{0.22} + T_w$$

$$T_{crit} = 18\left(\rho_v h_{fg} Q^{"}\left(\frac{\sigma}{\rho_w Q^{"2} d_{32}}\right)^{0.198}\right)^{\frac{1}{5.55}} + T_w$$

$$T_{\min} = 204.895 Q^{"0.066} \upsilon^{0.138} d_{32}^{\;-0.035} + T_w$$

$$T_{dfb} = 280.762 Q^{"0.087} \upsilon^{0.11} d_{32}^{\;-0.035} + T_w$$

$$q^{"}_{\;dfb} = 6100300 Q^{"0.588} \upsilon^{0.244}$$

$$q^{"}_{\;\min} = 3324400 Q^{"0.544} \upsilon^{0.324}$$

$$q^{"}_{\;crit} = 122.4 \rho_g h_{fg} Q^{"}\left(1 + 0.0118\left(\frac{\rho_w}{\rho_g}\right)^{0.25}\left(\frac{\rho_w Cp_w\left(T_w^{sat} - T_w\right)}{\rho_g h_{fg}}\right)\right)\left(\frac{\sigma}{\rho_w Q^{"2} d_{32}}\right)^{0.0198}$$

$$N_2 = \frac{q^{"}_{\;dfb} - q^{"}_{\;\min}}{\left(T_{dfb} - T_{\min}\right)^2}$$

$$N_1 = -2N_2\left(T_{\min}\right)$$

$$N_0 = q^{"}_{\;\min} - N_1\left(T_{\min}\right) - N_2\left(T_{\min}\right)^2$$

$T_s > T_{dfb}$: $$h = \frac{63.25\left(T_s - T_w\right)^{1.691} Q^{''0.264} d_{32}^{\;-0.062}}{\left(T_s - T_w\right)}$$

$T_s = T_{dfb}$: $$h = \frac{q''_{dfb}}{\left(T_s - T_w\right)}$$

$T_s < T_{dfb}; T > T_{\min}$: $$h = \frac{N_0 + N_1\left(T_s - T_w\right) + N_2\left(T_s - T_w\right)^2}{\left(T_s - T_w\right)}$$

$T_s = T_{\min}$: $$h = \frac{q''_{\min}}{\left(T_s - T_w\right)}$$

$T_s < T_{\min}; T > T_{crit}$: $$h = \frac{q''_{crit}}{\left(T_s - T_w\right)} - \frac{q''_{crit} - q''_{\min}}{\left(T_{crit} - T_{\min}\right)^3} \times$$

$$\frac{\left(T_{crit}^{\;3} - 3T_{crit}^{\;2}T_{\min} + 6T_{crit}T_{\min}\left(T_s - T_w\right) - 3\left(T_{crit} + T_{\min}\right)\left(T_s - T_w\right)^2 + 2\left(T_s - T_w\right)^3\right)}{\left(T_s - T_w\right)}$$

$T_s = T_{crit}$: $$h = \frac{q''_{crit}}{\left(T_s - T_w\right)}$$

$T_s < T_{crit}; T > T_{incip}$: $$h = \frac{0.0000187\left(T_s - T_w\right)^{5.55}}{\left(T_s - T_w\right)}$$

$T_s < T_{incip}$: $$h = \frac{Nu_w k_w}{d_{32}}$$

5.8 Tap information and composition of the two thermocouple blocks

Tap chemistry

	TiO_2	FeO	Al_2O_3	CaO	Cr_2O_3	MgO	MnO	SiO_2	V_2O_5	Total ****
Tap 37	85.42	13.25	0.71	0.04	0.11	0.89	1.51	1.01	0.45	104.02
Tap 38	84.21	13.92	0.75	0.04	0.15	1.26	1.45	0.94	0.45	102.94

Tap details

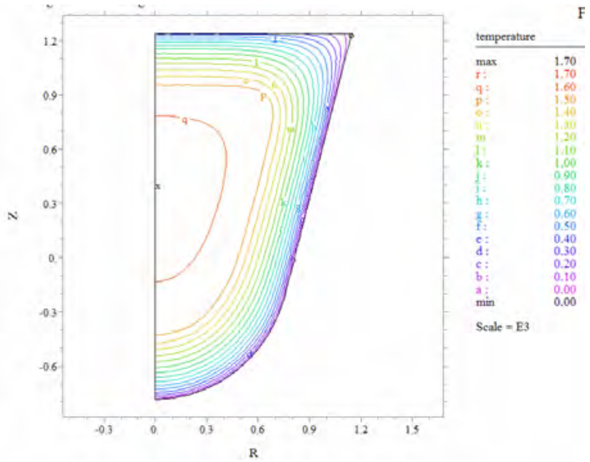
	Block mass (kg)	Tap rate (kg/min)	Tap temperature (˚C)
Tap 37	1,002	204.6	1669
Tap 38	1,365	345.7	1668

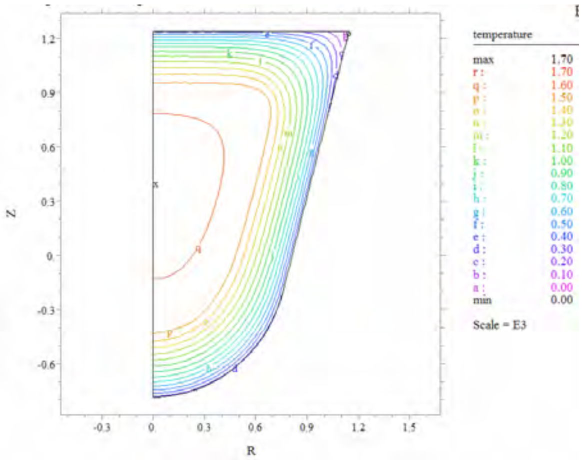
**** The total exceeds 100% because all titanium, including Ti^{3+},is reported as Ti^{4+} (TiO_2)

5.9 Cross sections water vs. air cooling

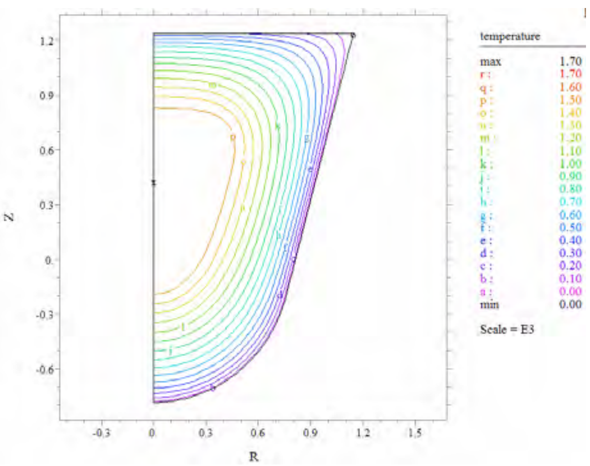
Water

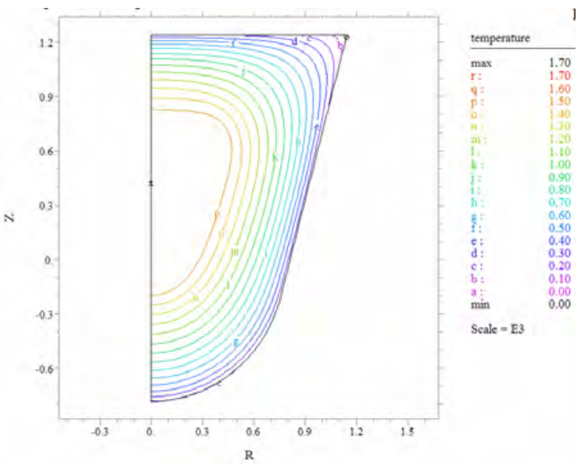
Air



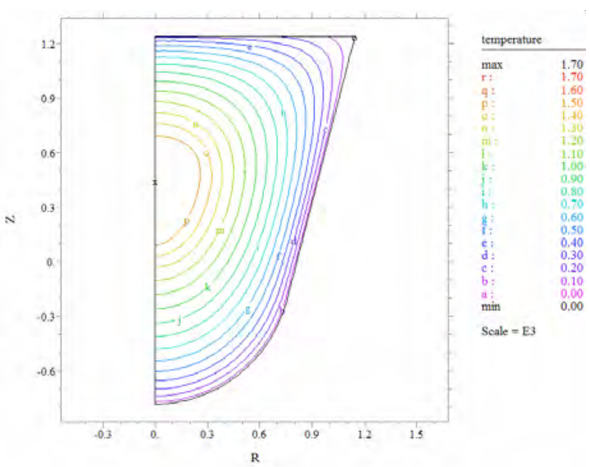


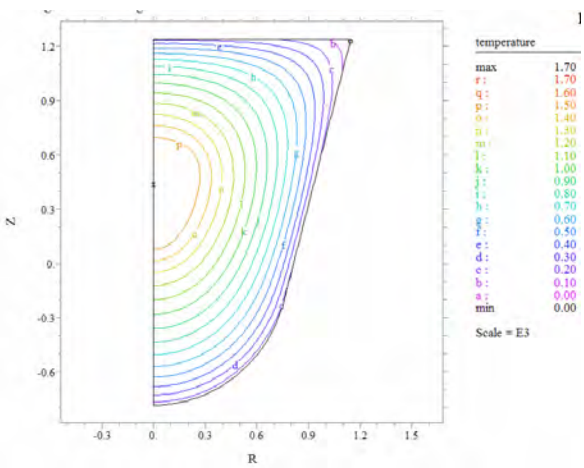
1 day





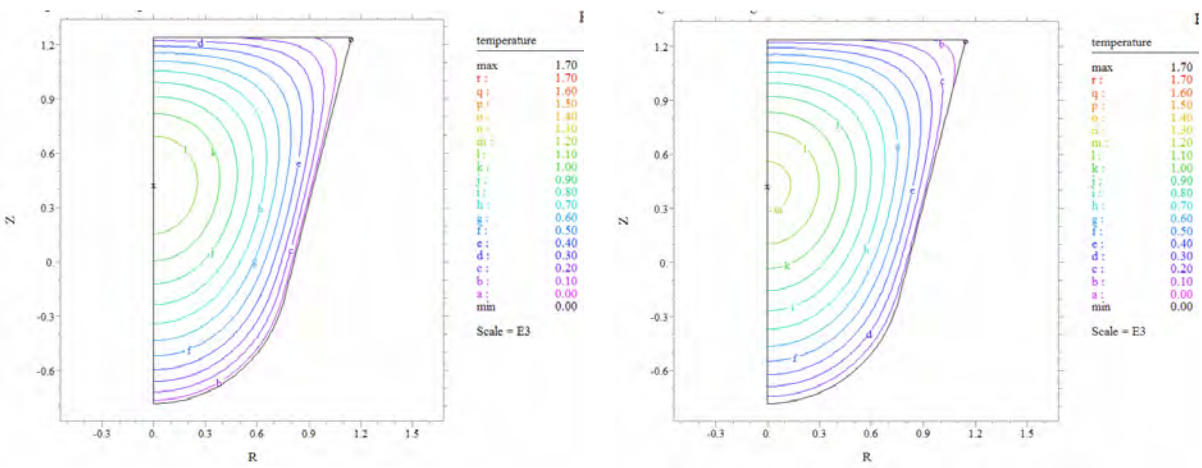
2 days



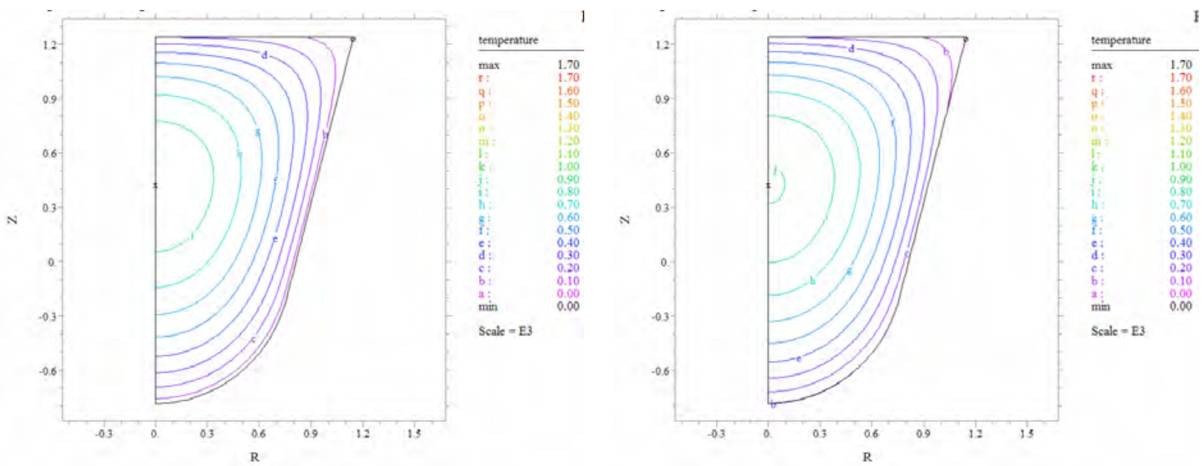


3 days

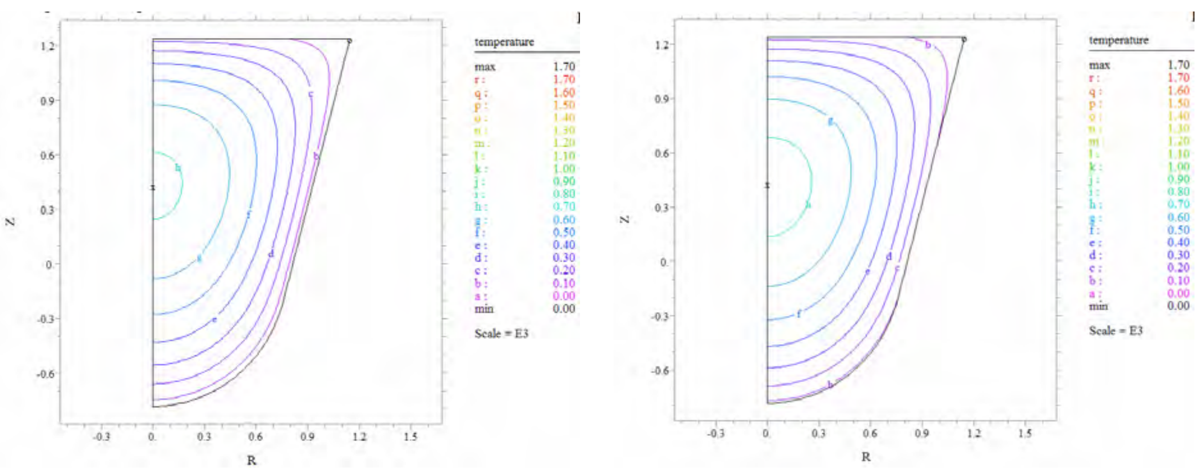
Water Air



4 days

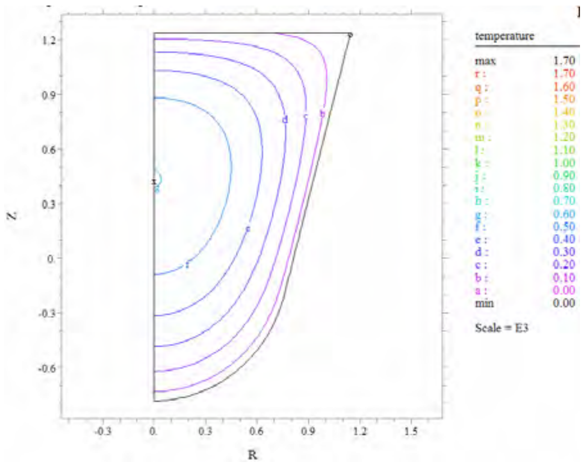


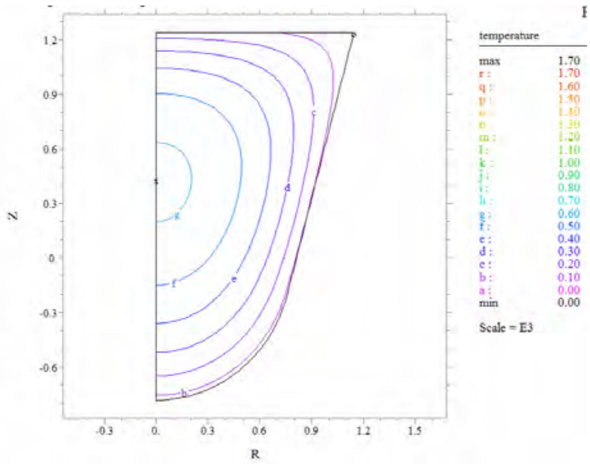
5 days



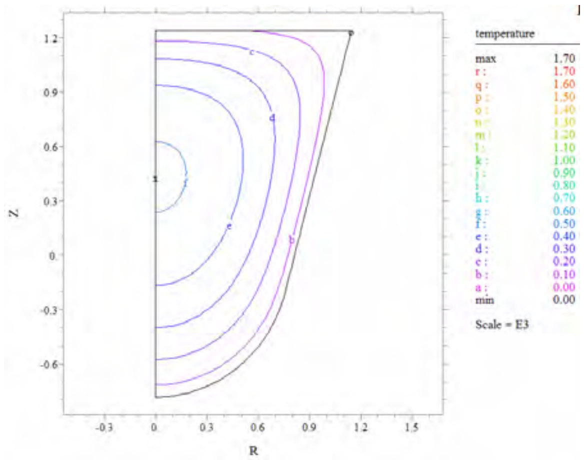
6 days

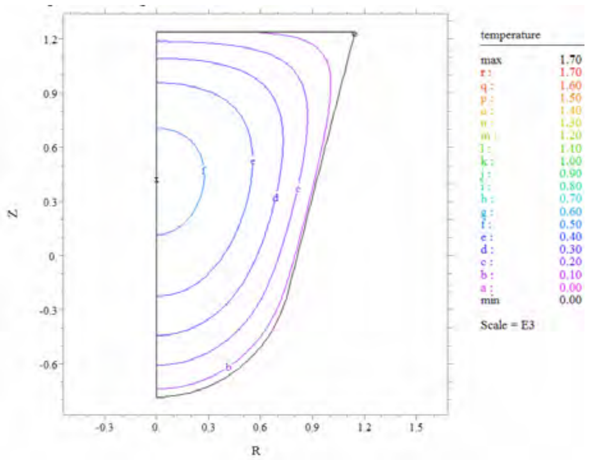
Water | Air



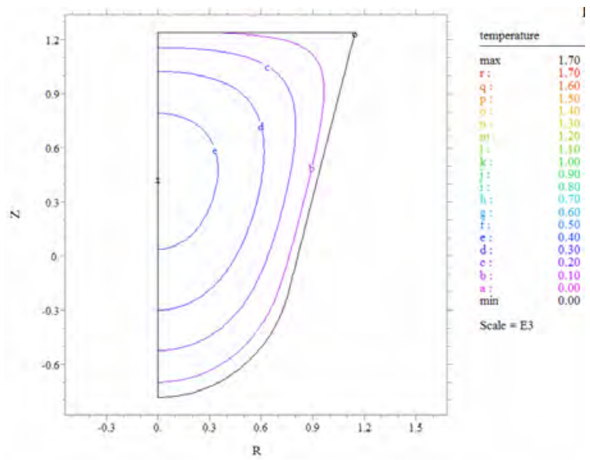


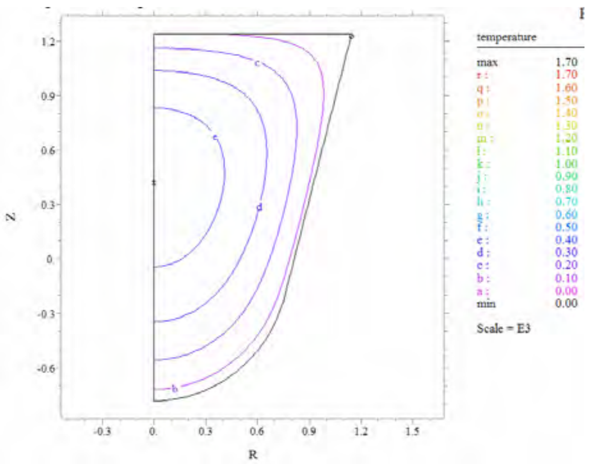
7 days





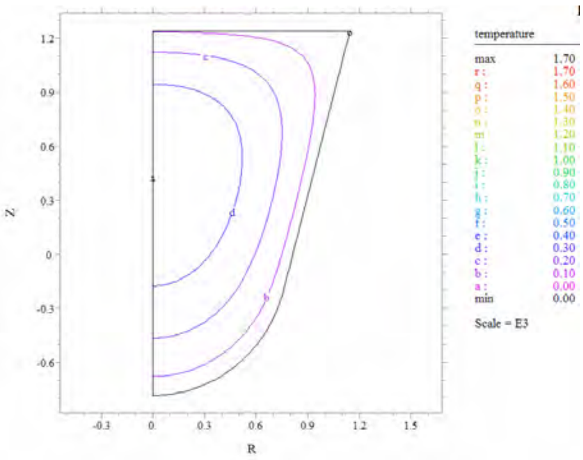
8 days



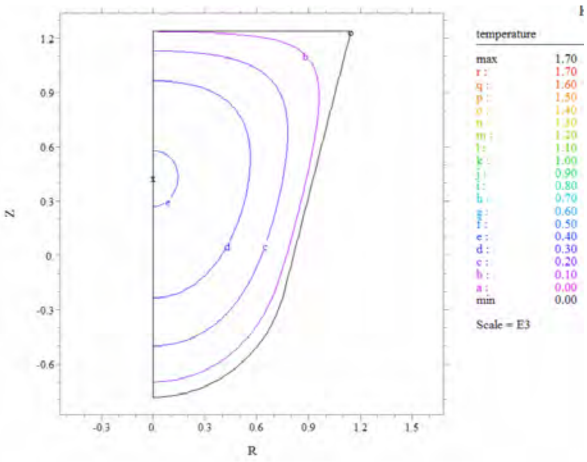


9 days

Water



Air



10 days

6 References

1 http://en.wikipedia.org/wiki/Titanium_dioxide. (Accessed 15 September 2006)

2 T. P. Battle, D. Nguyen and J.W. Reeves: "The Processing of titanium-containing ores." *The Paul E. Queneau International Symposium.* Vol. I: Fundamental Aspects., pp. 925-945, 1993.

3 H. Kotzé, D. Bessinger and J. Beukes: "Ilmenite smelting at Ticor SA." *The Journal of the South African Institute for Mining and Metallurgy*, March 2006, vol. 106, pp. 165-170.

4 PC Pistorius and C Coetzee: "Physicochemical aspects of titanium slag production and solidification." *Metallurgical and Materials Transactions Series B*, vol. 34B, pp. 581-588B (2003).

5 Bessinger, D., Geldenhuis, J.M.A. and Pistorius P.C.: "Phase changes in the decrepitation of solidified high titania slags." *Heavy Minerals 2005*, Society for Mining, Metallurgy and Exploration, 2005. pp. 213-219.

6 Bessinger, D., Geldenhuis, J.M.A., Pistorius, P.C., Mulaba, A., Hearne, G.: "The decrepitation of solidified high titania slags." *Journal of Non-Crystalline Solids*, vol. 282, no. 1, pp. 132–142 (2001).

7 J.P.R de Villiers, J. Göske and A. Tuling: "Disintegration in high grade titania slags: low temperature oxidation reactions and associated fracture mechanics of pseudobrookite." *Mineral Processing and Extractive Metallurgy (Trans. Inst. Min. Metall C)*, June 2005, vol. 114, pp. C73-C79.

8 M Gous: "An overview of the Namakwa Sands ilmenite smelting operations." *The Journal of the South-African Institute of Mining and Metallurgy*, June 2006, vol. 106, pp. 379-384.

9 P.C. Pistorius: "Ilmenite smelting – the basics." *The 6th International Heavy Minerals Conference 'Back to Basics'*. The Southern African Institute of Mining and Metallurgy, 2007. pp. 75-83.

10 J.H. Zietsman and P.C. Pistorius: "Process mechanisms in ilmenite smelting." *Journal of the South African Institute of Mining and Metallurgy*, vol. 104, pp. 653-660 (2004).

11 M. Guéguin and F. Cardarelli: "Chemistry and mineralogy of titania-rich slags. Part 1—hemo-ilmenite, sulphate, and upgraded titania slags." *Mineral processing and extractive metallurgy review*, vol. 28, pp. 1-58 (2007).

12 K. Borowiec, A.E. Grau, M. Guéguin and J.-F. Turgeon: "Method to upgrade titania slag and resulting product." United States Patent no. 5,830,420 (1998).

13 H. Elstad, J.M. Eriksen, A. Hildal, T. Rosenqvist, and S. Seim: "Equilibrium between titania slags and metallic iron." *The 6th International Heavy Minerals Conference 'Back to Basics'*. The Southern African Institute of Mining and Metallurgy, 2007. pp. 35-42.

14 G. Eriksson and A.D. Pelton: "Critical evaluation and optimization of the thermodynamic properties and phase diagrams of the MnO-TiO_2, MgO-TiO_2, FeO-TiO_2, Ti_2O_3-TiO_2, Na_2O-TiO_2 and K_2O-TiO_2 systems." *Metallurgical Transactions B*, vol. 24B, pp. 795-805 (1993).

15 G. Eriksson, A.D. Pelton, E. Woermann and E. Ender: "Measurement and thermodynamic evaluation of phase equilibria in the Fe-Ti-O system." *Berichte der Bunsengesellschaft für physikalische Chemie*, vol. 100, pp. 1839-1849 (1996).

16 J. Pesl and R.H. Eric: "High-temperature phase relations and thermodynamics in the iron-titanium-oxygen system." *Metallurgical and Materials Transactions B*, vol. 30B, pp. 695-705 (1999).

17 C.W. Bale, P. Chartrand, S.A. Degterov, G. Eriksson, K. Hack, R. Ben Mahfoud, J. Melançon, A.D. Pelton and S. Petersen: "FactSage Thermochemical Software and Databases." *Calphad*, vol. 26, pp. 189-228 (2002).

18 S. Jahanshahi, L. Zhang and D. Bessinger: "Development of MPE package and its application in ilmenite smelting." *Heavy Minerals 2003*. South African Institute of Mining and Metallurgy, 2003. pp. 119-126.

19 D.J. Fourie, J.J. Eksteen and J.H. Zietsman: "Calculation of FeO-TiO_2-Ti_2O_3 liquidus isotherms pertaining to high titania slags." *Journal of the South African Institute of Mining and Metallurgy*, vol. 105, pp. 695-710 (2005).

20 PC Pistorius: "Fundamentals of freeze lining behaviour in ilmenite smelting." *Journal of the South African Institute of Mining and Metallurgy*, vol. 103, pp. 509-514 (2003).

21 J.M.A. Geldenhuis and P.C. Pistorius. "The use of commercial oxygen probes during the production of high titania slags." *Journal of the South African Institute of Mining and Metallurgy*, vol. 99, pp. 41-47 (1999).

22 P.C. Pistorius and T. Motlhamme: "Oxidation of high-titanium slags in the presence of water vapour." *Minerals Engineering*, vol. 19, pp. 232 236 (2006).

23 I.E. Grey, L.M.D. Cranswick, C. Li, T.J. White, and L.A. Bursill: "New M_3O_5-anatase intergrowth structures formed during low-temperature oxidation of anosovite." *Journal of Solid State Chemistry*, vol. 150, pp. 128-138 (2000).

24 J.P. van Dyk, N.M. Vegter, C.P. Visser, T. de Lange, J.D. Winter, E.A. Walpole and J. Nell: "Beneficiation of titania slag by oxidation and reduction treatment." United States patent no. 6,803,024 (2004).

25 Errol G. Kelly, David J. Spottiswood: "Introduction to mineral processing." *Australian Mineral Foundation*, pp. 461, (1995).

26 P.C. Pistorius: "The relationship between FeO and Ti_2O_3 in ilmenite smelter slags." *Scandinavian Journal of Metallurgy*, 2002, vol. 31, pp. 120-125.

27 R.M. Viljoen, J.T Smit, I. du Plessis and V.Ser: "The development and application of in-bed compression breakage principles." *Minerals Engineering*, vol. 14, No. 5, pp. 465 – 471, 2001.

28 H Brundiek and F Poeschl: "A roller mill for cement and blast furnace slag in theory and practice." *37th IEEE Cement Industry Technical Conference*, June 1995, San Juan, Puerto Rico. pp. 197-223.

29 H.S. Carslaw and J.C. Jaeger: *Conduction of heat in solids,* second edition. Clarendon, 1959. pp. 282-286.

30 J.P. Holman. "Heat Transfer." *McGraw-Hill Book Co*, SI Metric Edition, pp. 635, (1989).

31 A. Hamasaiid, M.S. Dargusch, C.J. Davidson, S. Tovar, T. Loulou, F. Rezaï-Aria, and G. Dour: "Effect of mold coating materials and thickness on heat transfer in permanent mold casting of aluminum alloys." *Metallurgical and Materials Transactions A*, vol. 38A, pp. 1303-1316 (2007).

32 R. Le Goff, G. Poutot, D. Delaunay, R. Fulchiron, and E. Koscher: "Study and modeling of heat transfer during the solidification of semi-crystalline polymers." *International Journal of Heat and Mass Transfer*, vol. 48, pp. 5417–5430 (2005).

33 M. Bahrami, J.R. Culham, M.M. Yananovich and G.E. Schneider: "Review of thermal joint resistance models for nonconforming solid surfaces." *Applied Mechanics Reviews*, vol. 59, pp. 1-12 (2006).

34 M.G. Cooper, B.B. Mikic and M.M. Yovanovich: "Thermal contact conductance." *International Journal of Heat and Mass Transfer*, vol. 12, pp. 279-300 (1969).

35 J.J. Salgon, F. Robbe-Valloire, J. Blouet and J. Bransier: "A mechanical and geometrical approach to thermal contact resistance." *International Journal of Heat and Mass Transfer*, vol. 40, pp. 1121-1129 (1997).

36 J.P. Holman. "Heat Transfer." *McGraw-Hill Book Co*, SI Metric Edition, pp. 331-345, (1989).

37 W.P. Klinzing, J.C. Rozzi and I. Mudawar: "Film and transition boiling correlations for quenching of hot surfaces with water sprays." *J. Heat Treating*, (1992) vol. 9, pp. 91-103.

38 I. Mudawar and W.S. Valentine: "Determination of the local quench curve for spray-cooled metallic surfaces." *J. Heat Treating*, (1989), vol. 7, pp. 107-121.

39 T. Tran, C. Solnordal and C. Nexhip: "Determination of thermal conductivity of titania slags." *Unpublished CSIRO Minerals Report*, DMR-2229, June 2003.

40 Y. Lee and D. Deming: "Evaluation of thermal conductivity temperature corrections applied in terrestrial heat flow studies." *Journal of Geophysical Research*, vol. 103, pp. 2447-2454 (1998).

41 J.R. Smyth and T.C. McCormick: "Crystallographic data for Minerals", in *Mineral Physics and Crystallography, a handbook of physical constants*. American Geophysical Union, 1995. p. 3.

42 M.D. Lind and R.M. Housley: "Crystallization Studies of Lunar Igneous Rocks: Crystal Structure of Synthetic Armalcolite." *Science*, vol. 175, no. 4021, pp. 521-523 (1972).

43 W.D. Kingery, H.K. Bowen and D.R. Uhlmann: *Introduction to ceramics*, second edition. John Wiley & Sons, 1976. pp. 612-643.

44 H.J. Siebeneck, D.P.H. Hasselman, J.J. Cleveland and R.C. Bradt: "Effects of grain size and microcracking on the thermal diffusivity of $MgTi_2O_5$." *Journal of the American Ceramic Society*, vol. 60, pp. 336-338 (1977).

45 H.J. Siebeneck, D.P.H. Hasselman, J.J. Cleveland and R.C. Bradt: "Effects of microcracking on the thermal diffusivity of Fe_2TiO_5." *Journal of the American Ceramic Society*, vol. 59, pp. 241-244 (1976).