University of Pretoria etd – De Kock, D J (2005)

A

APPENDIX A: WATER MODEL VALIDATION STUDY

A-1 INTRODUCTION

An important part of the optimisation is to trust the accuracy of the numerical model. For this purpose a water model validation study was done. This work was published in Mechanical Technology [65]. The details of this study are given in this appendix. In physical modelling, water is normally used to simulate molten steel in perspex models of tundishes. In this modelling, emphasis has been placed on the matching of Reynolds (Re) and Froude (Fr) similarity criteria [17]. When water at room temperature is used as the modelling fluid, matching of both Reynolds and Froude numbers require the use of full-scale models. In reduced-scale models, either Reynolds or Froude numbers can be satisfied. A full-scale water model of the Columbus Stainless Steel tundish was commissioned at the University of Pretoria in 1999 [66] therefore having the advantage of having both Reynolds and Froude number similarity satisfied. The tundish model is constructed from 10mm-thick perspex sheets and supported by a steel frame. The water flow network provides the option to recirculate the water from the tundish outlet to the reservoir, or to divert it to a drain (see Figure A-1).

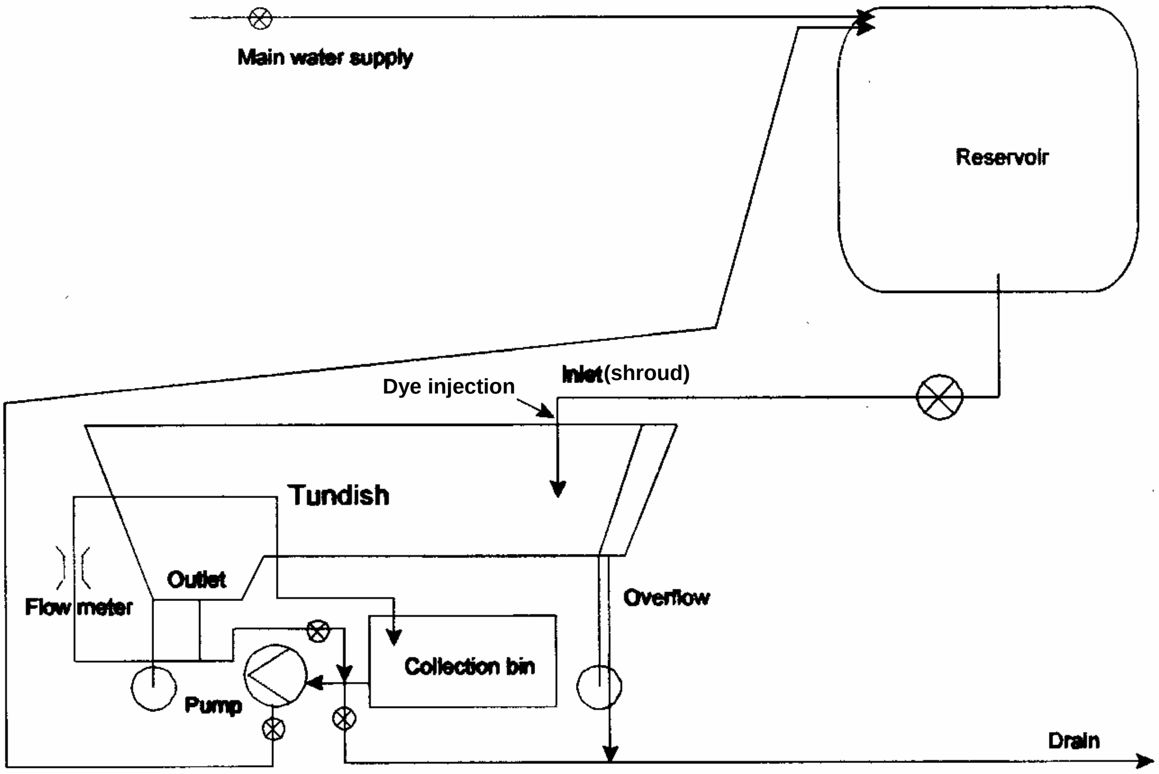

Figure A-1: Schematic layout of the water model

In water modelling studies, a pulse of tracer such as dye or a salt solution is injected in the incoming water stream while its concentration at the outlet is measured as a function of time. The plot of the exit concentration against time is known as the residence time distribution (RTD) curve. The RTD of a fluid in the tundish is used to characterise the fluid flow. The RTD is commonly used to determine the volumes of plug flow, mixed flow and dead regions in the tundish. The use of dye as tracer also facilitates the visual observation of the flow patterns through the tundish and helps in identifying the locations of slow moving or turbulent and well-mixed regions.

In mathematical modelling, the turbulent Navier-Stokes equations are solved with appropriate boundary conditions to yield detailed information on velocity, pressure and turbulence fields in the tundish. Such models can also be used to theoretically determine the residence time distribution of fluid in the tundish by releasing a passive scalar at the inlet and tracking its concentration distribution through time at the outlet.

A-2 EXPERIMENTAL SETUP

The experimental process involves achieving a stable operating level in the tundish for a period which is at least 3 times the average residence time before an experiment is initiated by injecting the dye. A water sample is obtained from a sampling tube inside the outlet pipe. The water sample is fed through a spectrophotometer where the absorbance at a specified wavelength is measured. The absorbance of the sample is referenced to the absorbance of clean water. A pre-measured amount of the dye solution is injected into the shroud (inlet to the tundish) as a pulse over three to five seconds. The dye input is called the tracer and the mixing of the tracer in the water changes the absorbance of the water.

A-2.1 SPECTROPHOTOMETER

A Novaspec II visible spectrophotometer [67] was used to determine the residence time distribution curve of the tracer at the outlet of the water model. The spectrophotometer provides measurement of both light absorbance and light transmission in the visible wavelength range (325–900nm). The diagram in Figure A-2 shows in simplified form the optical system of the instrument. The light output from the lamp passes through a slit into the monochromator where a collimating mirror converts the light into a parallel beam. The light then passes to the diffraction grating to produce light of the selected wavelength. This monochromatic light then passes through the stray light filters and the monochromator outlet slit and then through the cuvette with the water sample to the solid-state detector unit.

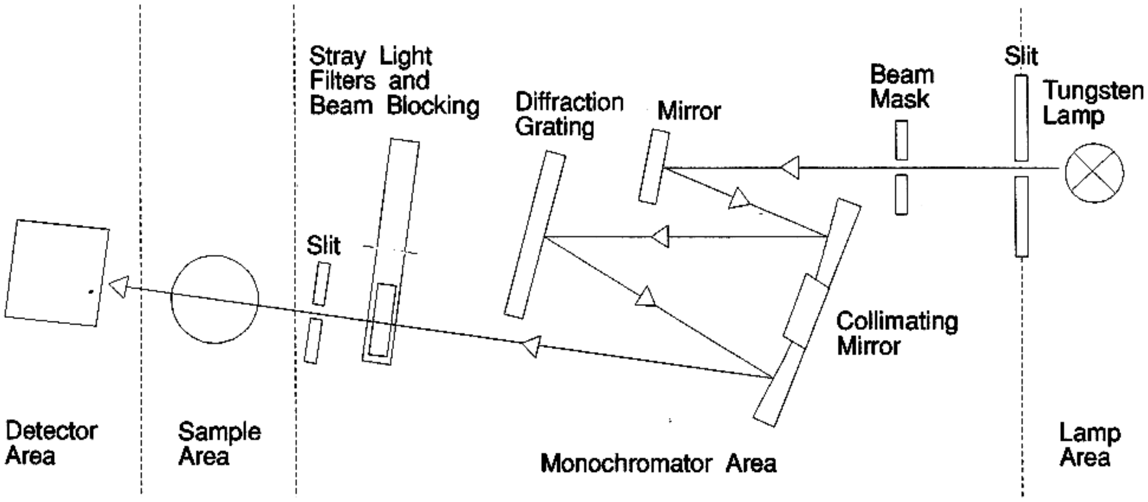

Figure A-2: The spectrophotometer's optical system [67]

The relationship between the concentration of the sample and its absorbance is linear, and hence absorbance is widely used experimentally. The absorbance (A) is calculated from the logarithm of the sample transmittance (T) by applying the Beer-Lambert law A=log 100/T. The result is displayed in units of absorbance. In order to calculate the transmittance, the instrument measures the amount of light at the specified wavelength that has passed through the sample and compares it with that which has passed through the reference. The dye used in the experiment has a wavelength of 435nm.

A-2.2 DIGITAL CAMCORDER

A Sony TRV110 digital camcorder is used to capture the experiment digitally on video. This digital video is then transmitted via an IEEE 1394 Firewire interface to a PC for further digital editing and processing.

A-2.3 ULTRASONIC FLOW METER

The Altosonic Ultrasonic Flow Meter [66] is a portable flow meter that can be readily attached to existing pipelines. Measurement is performed without any obstruction to the flow of the medium and no alterations to the pipe section involved are necessary, thus no additional pressure loss will occur. A sound wave that is sent in the direction of the flow through a medium will travel faster than one that is sent in the opposite direction. This principle is used in the ultrasonic transit time flow meter.

A-3 RESULTS

A-3.1 RESIDENCE TIME DISTRIBUTION (RTD)

The RTD curve of the experiment (repeated four times) is compared to the RTD curves obtained by the CFD code in Figure A-3 for an empty tundish. Different turbulence modelling options were tested in the CFD model and the one shown (RNG k-ε turbulence model) gave the best results. The experimental set-up was emulated by a 2 sec pulse injection in the CFD model to obtain the RTD.

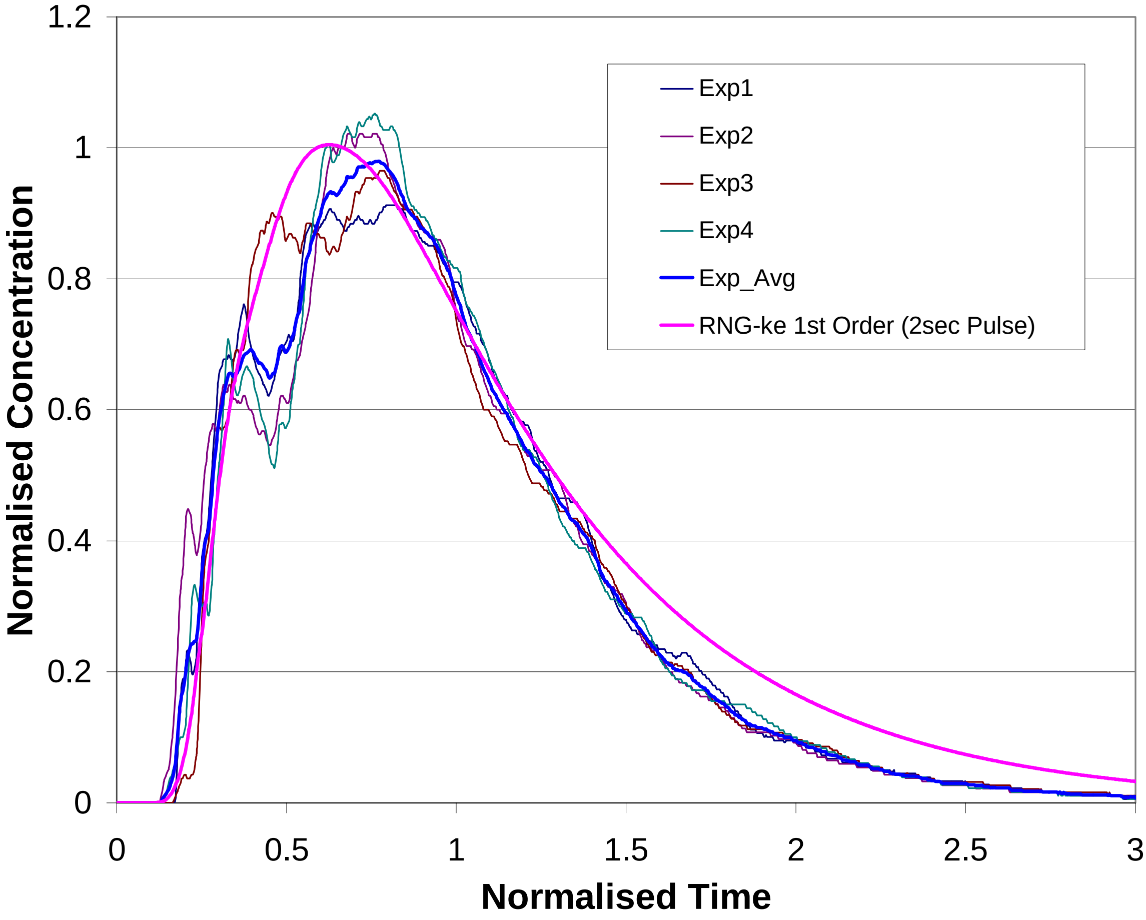

Figure A-3: Comparison of RTD curves

From the RTD curve the relevant RTD data were extracted to compare the experimental and CFD data. These data are given in Table A-1. Generally, good agreement is found between the experimental and CFD data with the Minimum Residence Time (MRT) being the only value that is not predicted to within 15%. This MRT is however very sensitive to the time that the injection took and the sensitivity of the spectrophotometer.

Table A-1: Comparison of RTD data of the water model and CFD model

	Water model	CFD	% Difference
MRT	0.128	0.154	20.3%
Dispersed Plug Flow (V_{dpv})	0.450	0.391	13.1%
Mixed Flow (V_{mv})	0.406	0.452	11.3%
Dead Volume (V_{dv})	0.144	0.156	8.3%

A-3.2 COMPARISON OF FLOW PATTERNS

Three frames captured by the digital video and given in Figure A-4 show the dye pattern at different times after the dye has been injected. The first frame shows the dye pattern just after the injection of the dye. In the second frame the dye started to move towards the outlet in a uniform pattern. In the last frame the short circuiting to the outlet can be clearly seen.

A comparison between the experimental and numerical results is given in Figure A-5. The numerical results show the scalar concentration on the centre line and side of the tundish. In general the patterns are similar for the experimental and numerical results. The numerical results show that there is significant variation between the centre and side of the tundish.

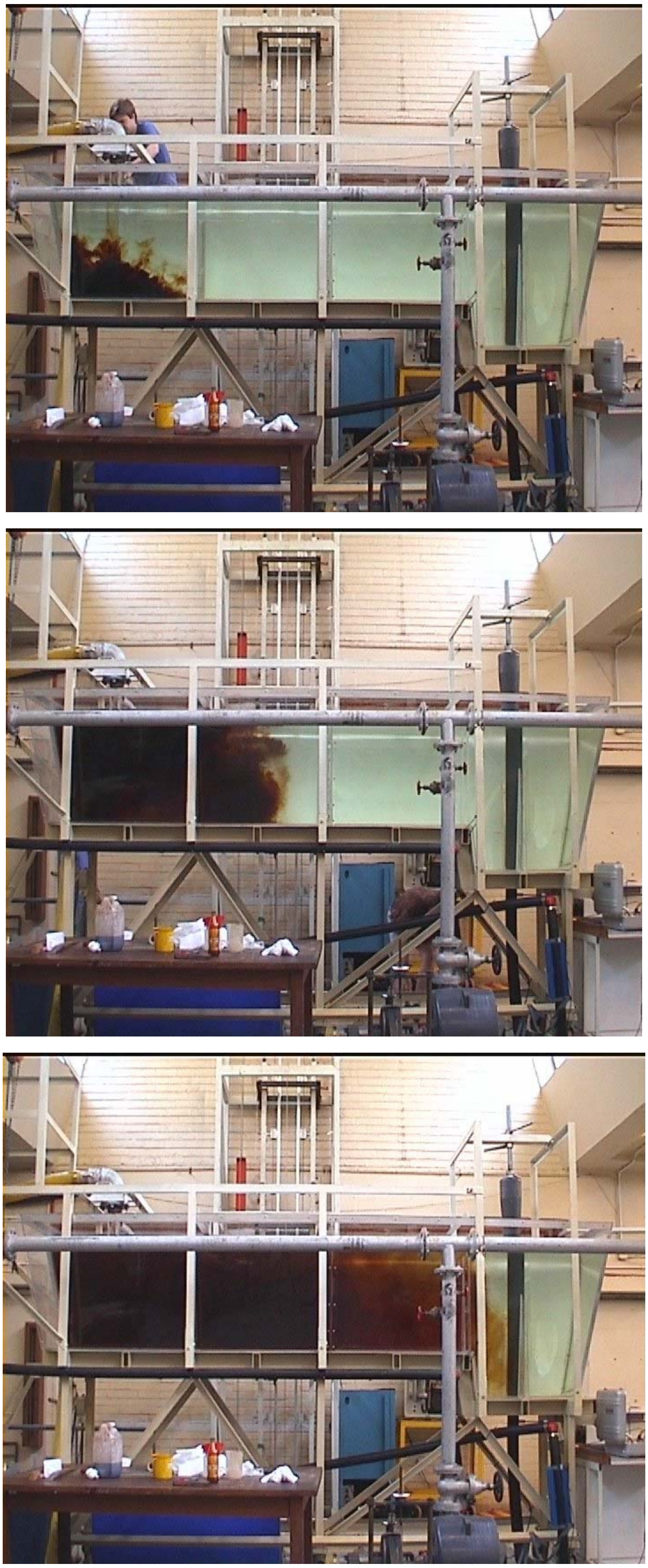

Figure A-4: Dye flow patterns at different times after dye injection

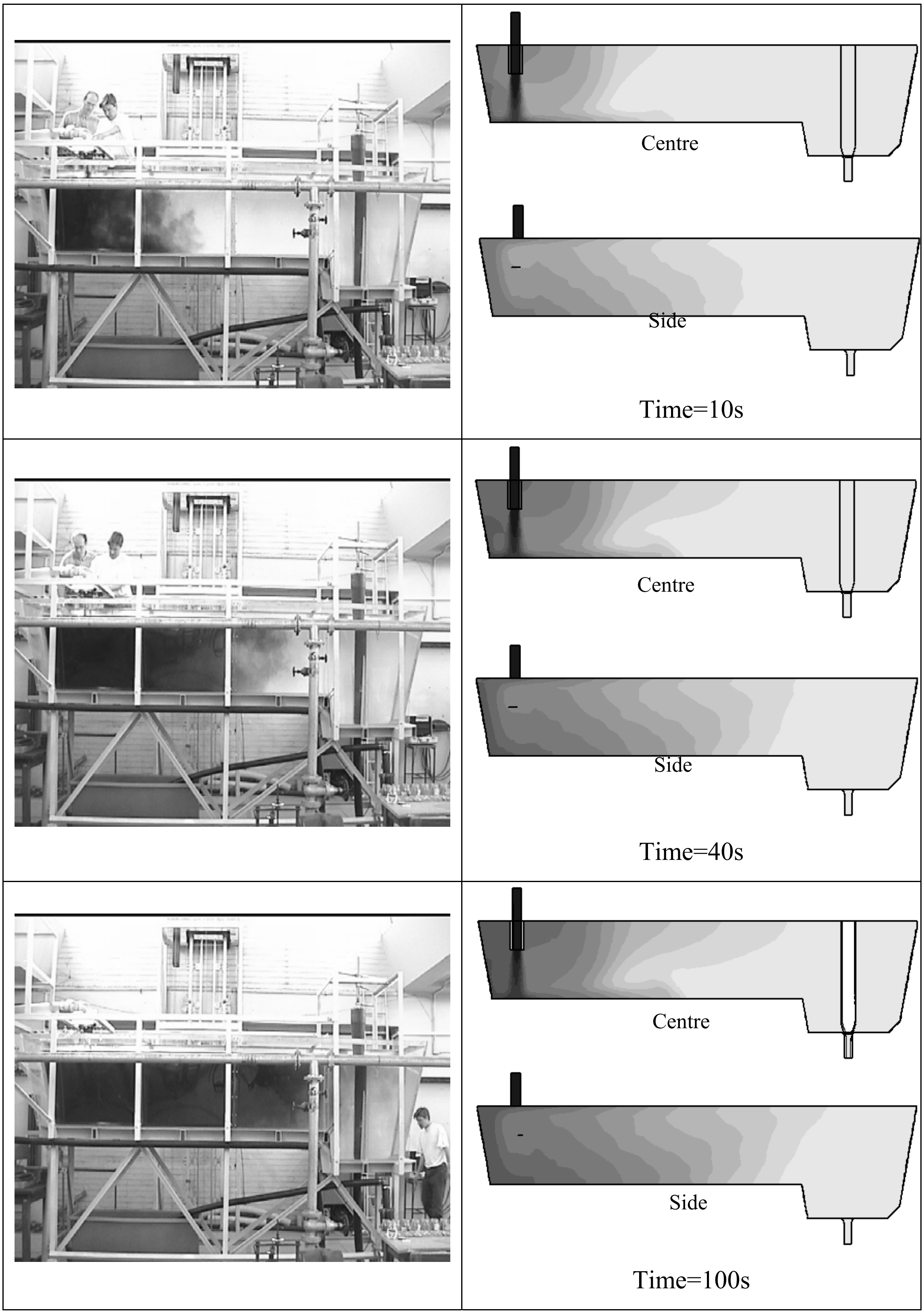

Figure A-5: Comparison between the experimental (left) and numerical (right) results. The numerical results show the scalar concentration.

University of Pretoria etd – De Kock, D J (2005)

A-4 CONCLUSION AND FUTURE WORK

There appears to be a good correlation between the experimental results and the CFD results for this simple set-up. The diffusion of the tracer during the experiment could also not be quantified.

B

APPENDIX B: PLANT TRIAL VALIDATION STUDY

A validation study between experimental plant data and the CFD modelling was carried out to ensure that the CFD modelling techniques used are calibrated. The validation was done on the V1-V2 continuous caster of ISCOR Vanderbijlpark, South Africa. The configuration used for the trial is shown in Figure B-1.

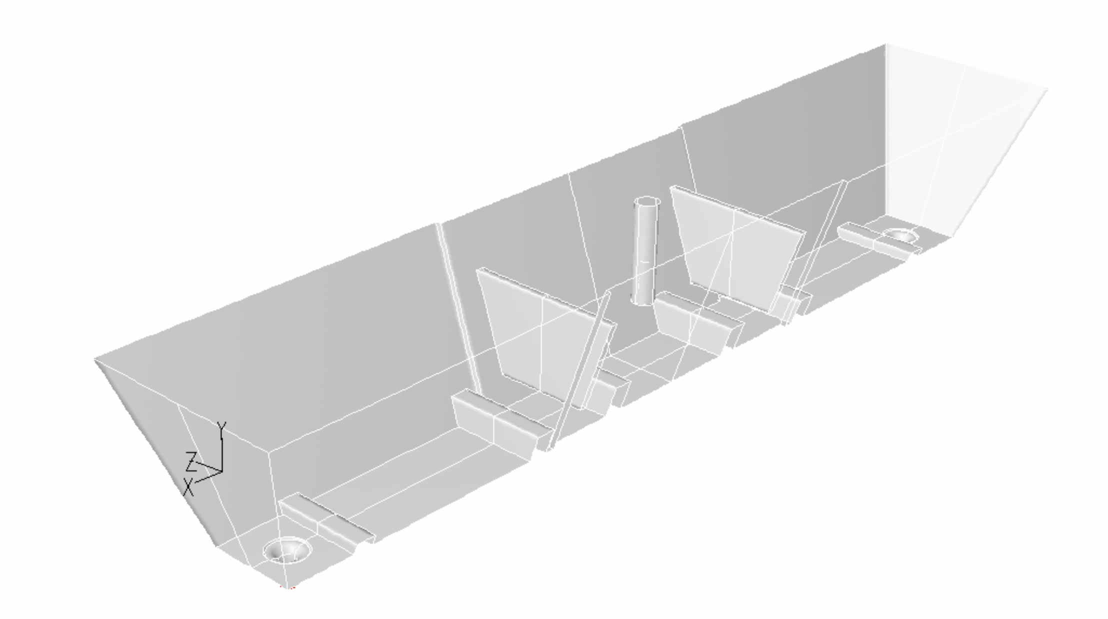

Figure B-1: Tundish configurations for validation study

Table B-1 gives the details of the CFD model used in the validation study.

Table B-1: Details of CFD model used in validation study

Description	Setting
Model	Non-isothermal steady state flow, turbulence and heat transfer, transient species concentration
Turbulence model	Standard k-ε
Discretisation scheme	2nd order for pressure 2nd order upwind for all other equations
Liquid Steel property:	
Reference density	7026.8kg/m^3
Reference temperature	1800K
Thermal conductivity	52.5W/m·K
Viscosity	0.0053kg/m·s
Thermal expansion coefficient	1.197×10^{-4}K^{-1}
Inlet boundary condition:	
Mass flow	16.67kg/s (quarter model)
Inlet temperature	1856.15K
Inlet turbulence intensity	10%
Inlet hydraulic diameter	0.1025m
Outlet boundary condition:	Zero gradient for all equations normal to boundary
Wall boundary condition:	
Bottom effective heat transfer coefficient	1.25W/m^2·K
Side effective heat transfer coefficient	2.76W/m^2·K
Slag effective heat transfer coefficient	11.4W/m^2·K
Free stream temperature	298.15K
Remaining walls (dam, baffles, etc.)	Adiabatic

A copper trace experiment was conducted on-site, which involved the introduction of copper next to the shroud and then periodically taking samples from the mould. The copper concentration in the samples was measured using a photo-spectrometer [28, 29]. A comparison of the CFD-predicted RTD curve and the measured points is shown in Figure B-2. It can be seen in this figure that the plant trial and CFD are in good agreement. The RTD data extracted from these curves are given in Table B-2.

It is mainly the dead flow volume that differs significantly with the plant trial results. The comparison is however fairly good taking into account all the uncertainties associated with the plant trial. For example, the path of the tracer differs between the plant trial and numerical model, as the copper is introduced into the tundish next to the shroud on-site and in the CFD model the tracer is introduced in the shroud.

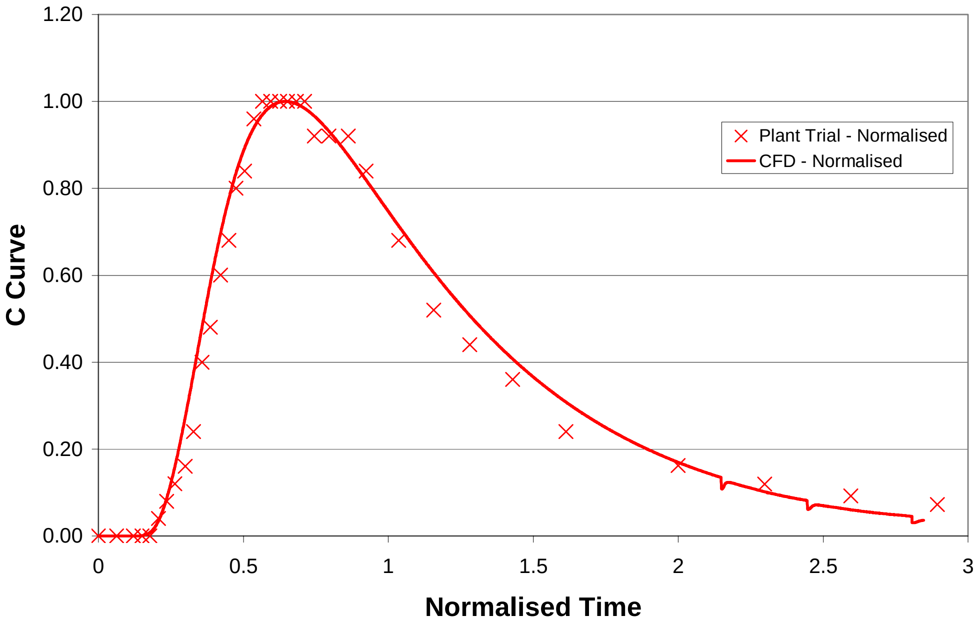

Figure B-2: Comparison of RTD curves of the plant trial and CFD model

Table B-2: : Comparison of RTD data of the plant trial and CFD model

	Plant Trial	CFD	% Error
Plug flow volume [%]	20.7	18.2	11.9
Mixed flow volume [%]	74.5	75.7	1.6
Dead flow volume [%]	4.8	6.1	27.1

APPENDIX C: MODIFIED DYNAMIC-Q PROGRAM

The Dynamic-Q program was modified to be able to link to Fluent and Gambit. The complete program listing in Fortran is given in this appendix. The modification made to the programme is the automatic linking with Fluent and Gambit as well as the automatic extraction of the data (refer to system call in especially fch77.for).

driver77.for

```
      PROGRAM DRIVER77
C**********************************************************************C
C                                                                      C
C      DYNAMIC-Q ALGORITHM FOR CONSTRAINED OPTIMIZATION                C
C                                                                      C
C           Sample driver for the subroutine DynQ77                    C
C                                                                      C
C                                                                      C
C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!C
C                                                                      C
C  User specified subroutines:                                         C
C                                                                      C
C  The objective function and constraint functions must be specified   C
C  in subroutine FCH. Expressions for the respective gradient vectors  C
C  must be specified in subrountine GradFCH.                           C
C                                                                      C
C  {The user may compute gradients by finite differences if necessary  C
C   - see example code in GradFCH}                                     C
C                                                                      C
C  Side constraints should not be included as inequality constraints   C
C  in the above subroutines, but entered in the relevant section below.C
C                                                                      C
C  Further quantities to be specified are the number of variables N,   C
C  starting point X, number of inequality NP, equality NQ, lower NL    C
C  and upper limits NU. Also specify the move limit DML and            C
C  convergence tolerances on step size and function value XTOL and     C
C  FTOL and maximum number of iterations KLOOPMAX.                     C
C                                                                      C
C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!C
C                                                                      C
C  The application of the code is illustrated here for the very simple C
C  but general example problem (Hock 71):                              C
C                                                                      C
C       minimize  F(X) = X(1)*X(4)*(X(1)+X(2)+X(3))+X(3)               C
C  such that                                                           C
C                 C(X) = 25-X(1)*X(2)*X(3)*X(4) <= 0                   C
C       and                                                            C
C                 H(X) = X(1)**2+X(2)**2+X(3)**2+X(4)**2-40 = 0        C
C                                                                      C
C       and side constraints                                           C
C                                                                      C
C                 1 <= X(I) <= 5 , I=1,2,3,4                           C
C                                                                      C
C  The remainder of the code is self-explanatory. Read the comments    C
```

```
C  below for the specification of initial data and parameter settings. C
C                                                                      C
C**********************************************************************C
C
      IMPLICIT REAL*8 (A-H,O-Z),INTEGER(I-N)
      PARAMETER (NMAX=300,NLMAX=50,NUMAX=50)
      LOGICAL PRNCONST,FEASIBLE
      DIMENSION X(NMAX)
      DIMENSION NLV(NLMAX),NUV(NUMAX),V_LOWER(NLMAX),V_UPPER(NUMAX)
       DIMENSION DELTX(NMAX),DML(NMAX)
C
C*****OPEN OUPUT FILES*************************************************C
C
      OPEN (9,FILE='Lfopc.out')
      OPEN (10,FILE='Approx.out')
      OPEN (11,FILE='DynamicQ.out')
C
C*****SPECIFY NUMBER OF VARIABLES N ***********************************C
C                                                                      C
      N=6
C
C*****SPECIFY STARTING POINT (INITIAL GUESS) : X(I), I=1,N ************C
C                                                                      C
C      PI=4.D0*DATAN(1.0D0)
C
c      DO I=1,N
c            X(I)=8.D-1
c      END DO

      X(1)=650.
      X(2)=400.
      X(3)=50.
      X(4)=50.
      X(5)=20.
      X(6)=20.

C                                                                      C
C*****SPECIFY NUMBER OF INEQUALITIES NP (NP=0 IF NONE)****************C
C     (EXCLUDING SIDE CONSTRAINTS WHICH SHOULD BE SEPARATELY           C
C     SPECIFIED BELOW)                                                 C
C                                                                      C
      NP=1
C                                                                      C
C*****SPECIFY NUMBER OF EQUALITIES NQ   (NQ=0 IF NONE)****************C
C                                                                      C
      NQ=0
C
C*****SPECIFY SIDE CONSTRAINTS*****************************************C
C
C      Indicate the number of lower limits NL and upper limits NU
C            (0<=NU<=N and 0<=NU<=N)
      NL=N
      NU=N
C
C      Specify LOWER limits with V_LOWER(j) specifying the VALUE of
C      the limit and NLV(j)=I the associated VARIABLE X(I), j=1,NL
      IF (NL.GT.0) THEN

      DO I=1,NL
           NLV(I)=I
      END DO

c      DO I=1,NL
c            V_LOWER(I)=-90.0
c      END DO

       V_LOWER(1)=400.
       V_LOWER(2)=300.
```

```
       V_LOWER(3)=40.
       V_LOWER(4)=0.
       V_LOWER(5)=0.
       V_LOWER(6)=0.

      END IF

C      Specify UPPER limits with V_UPPER(j) specifying the VALUE of
C      the limit and NUV(j)=I the associated VARIABLE X(I), j=1,NU
      IF (NU.GT.0) THEN

      DO I=1,NU
            NUV(I)=I
      END DO

c      DO I=1,NU
c            V_UPPER(I)=0.0
c      END DO

       V_UPPER(1)=1200.
       V_UPPER(2)=600.
       V_UPPER(3)=10000.
       V_UPPER(4)=100.
       V_UPPER(5)=100.
       V_UPPER(6)=40.

      END IF
C
C*****GIVE THE CONVERGENCE TOLERANCES ON FUNC AND STEPNORM VALUES******C
C
      FTOL=1.D-8

      XTOL=1.D-5
C
C*****SPECIFY THE MOVE LIMIT DML***************************************C
C
c      DO I=1,N
c            DML(I)=30.0
c      END DO

       DML(1)=200.
       DML(2)=100.
       DML(3)=10.
       DML(4)=25.
       DML(5)=25.
       DML(6)=10.

C
C*****GIVE THE MAXIMUM NUMBER OF ITERATIONS****************************C
C
      KLOOPMAX=99
C
C*****PRINT CONSTRAINTS TO SCREEN AT END OF PROGRAM?******************C
C            YES: .TRUE.        OR            NO: .FALSE.
C
      PRNCONST=.TRUE.
C
C*****SPECIFY SIZE OF FINITE DIFFERENCE STEP SIZE*********************C
C
c      DO I=1,N
c            DELTX(I)=10.0
c      END DO

      DELTX(1)=100.
      DELTX(2)=50.
      DELTX(3)=5.
      DELTX(4)=12.
      DELTX(5)=12.
```

University of Pretoria etd – De Kock, D J (2005)

```
      DELTX(6)=5.

C
C*****FEASABLE LIMIT TO CHECK CONVERGENCE******************************C
C     (AMOUNT AN CONSTRAINT CAN BE VOILATED
C
      FEASLIM=1.D-1
C
C
C*****CALL SUBROUTINE DYNQ77*******************************************C
      CALL DYNQ77(N,X,NP,NQ,NL,NU,NLV,NUV,V_LOWER,V_UPPER,
     *      FTOL,XTOL,DML,KLOOPMAX,PRNCONST,DELTX,FEASLIM)

      STOP
      END PROGRAM DRIVER77
```

DynQ77.for

```
      SUBROUTINE DYNQ77(N,X,NP,NQ,NL,NU,NLV,NUV,V_LOWER,V_UPPER,
     *      FTOL,XTOL,DML,KLOOPMAX,PRNCONST,DELTX,FEASLIM)
C**********************************************************************C
C                                                                      C
C      DYNAMIC-Q ALGORITHM FOR CONSTRAINED OPTIMIZATION                C
C          GENERAL MATHEMATICAL PROGRAMMING CODE (FORTRAN 77)         C
C          -------------------------------------                       C
C                                                                      C
C                    Programmed by A.M. HAY                            C
C         Multidisciplinary Design Optimization Group (MDOG)           C
C   Department of Mechanical Engineering, University of Pretoria       C
C                       June 2000                                      C
C                                                                      C
C                    UPDATED : 18 August 2000                          C
C                                                                      C
C       Specific modification made to code for implementation into     C
C                       Fluent by DJ de Kock 2003                      C
C                                                                      C
C  This code is based on the Dynamic-Q method of Snyman documented     C
C  in the paper "THE DYNAMIC-Q OPTIMIZATION METHOD: AN ALTERNATIVE     C
C  TO SQP?" by J.A. Snyman and A.M. Hay. Technical Report, Dept Mech.  C
C  Eng., UP.                                                           C
C                                                                      C
C                   BRIEF DESCRIPTION                                  C
C                   -----------------                                  C
C                                                                      C
C   Dynamic-Q solves inequality and equality constrained optimization  C
C   problems of the form:                                              C
C                                                                      C
C                    minimize F(X)  ,  X={X(1),X(2),...,X(N)}          C
C     such that                                                        C
C                     Cj(X) <= 0               j=1,2,...,NP            C
C     and                                                              C
C                     Hj(X) =  0               j=1,2,...,NQ            C
C     with lower bounds                                                C
C         CLj(X) = V_LOWER(j)-X(NLV(j)) <= 0   j=1,2,...,NL            C
C     and upper bounds                                                 C
C         CUj(X) = X(NUV(j))-V_UPPER(j) <= 0   j=1,2,...,NU            C
C                                                                      C
C  This is a completely general code - the objective function and the  C
C  constraints may be linear or non-linear. The code therefore solves  C
C  LP, QP and NLP problems.                                            C
C                                                                      C
C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!C
C                                                                      C
C  Arguments of DynQ77 subroutine:                                     C
```

```
C                                                                      C
C  NAME    DIMENSION   TYPE      DESCRIPTION                           C
C  N                   INTEGER   Number of variables                   C
C  X       X(N)        DBL PREC  Starting point                        C
C  NP                  INTEGER   Number of inequality constraints      C
C  NQ                  INTEGER   Number of equality constraints        C
C  NL                  INTEGER   Number of lower bounds                C
C  NU                  INTEGER   Number of upper bounds                C
C  NLV     NLV(NL)     INTEGER   See description below                 C
C  NUV     NUV(NU)     INTEGER   See description below                 C
C  V_LOWER V_LOWER(NL) DBL PREC  See description below                 C
C  V_UPPER V_UPPER(NL) DBL PREC  See description below                 C
C  FTOL                DBL PREC  Termination tolerance on func. value  C
C  XTOL                DBL PREC  Termination tolerance on step size    C
C  DML                 DBL PREC  Move limit                            C
C  KLOOPMAX            INTEGER   Maximum number of iterations          C
C  PRNCONST            LOGICAL   Controls printing of constraints to   C
C                                screen at end of program              C
C  DELTX   DELTX(N)    DBL PREC  Size of finite difference             C
C  DML     DML(N)      DBL PREC  Move limit for each variable          C
C  FEASLIM             DBL PREC  Amount a constraint can be violated   C
C                                                                      C
C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!C
C                                                                      C
C      Specify LOWER limits with V_LOWER(j) specifying the VALUE of    C
C      the limit and NLV(j)=I the associated VARIABLE X(I), j=1,NL     C
C      Specify UPPER limits with V_UPPER(j) specifying the VALUE of    C
C      the limit and NUV(j)=I the associated VARIABLE X(I), j=1,NU     C
C                                                                      C
C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!C
C                                                                      C
C  The application of the code is illustrated here for the very simple C
C  but general example problem (Hock 71):                              C
C                                                                      C
C       minimize  F(X) = X(1)*X(4)*(X(1)+X(2)+X(3))+X(3)               C
C  such that                                                           C
C                 C(X) = 25-X(1)*X(2)*X(3)*X(4) <= 0                   C
C       and                                                            C
C                 H(X) = X(1)**2+X(2)**2+X(3)**2+X(4)**2-40 = 0        C
C                                                                      C
C       and side constraints                                           C
C                                                                      C
C                 1 <= X(I) <= 5 , I=1,2,3,4                           C
C                                                                      C
C                                                                      C
C  The remainder of the code is self-explanatory. Read the comments    C
C  below for the specification of initial data and parameter settings. C
C                                                                      C
C**********************************************************************C
C
      IMPLICIT REAL*8 (A-H,O-Z),INTEGER(I-N)
      DIMENSION X(N),DELX(N),GF(N),GC_V(N),GH_V(N)
      DIMENSION C(NP),GC(NP,N),BCURV(NP)
      DIMENSION H(NQ),H_A(NQ),GH(NQ,N),CCURV(NQ)
      DIMENSION NLV(NL),NUV(NU),V_LOWER(NL),V_UPPER(NU)
      DIMENSION C_A(NP+NL+NU+N)
      DIMENSION X_H(0:KLOOPMAX,N),F_H(0:KLOOPMAX)
      DIMENSION C_H(0:KLOOPMAX,NP+NL+NU+N),H_H(0:KLOOPMAX,NQ)
      LOGICAL PRNCONST,FEASIBLE
      DIMENSION DELTX(N),DML(N)

C
C######################################################################C
C**********************************************************************C
C      MAIN PROGRAM FOLLOWS:
C**********************************************************************C
C######################################################################C
C
```

```
C*****INITIALIZE OUTPUT************************************************C

      WRITE (10,'(/1X,"DYNAMICQ OUTPUT FILE")')
      WRITE (10,'(1X,"--------------------")')
      WRITE (10,'(/1X,"Number of variables [N]=",I3)'), N
      WRITE (10,'(1X,"Number of inequality constraints [NP]=",I3)'),NP
      WRITE (10,'(1X,"Number of equality constraints [NQ]=",I3)'),NQ
      WRITE (10,'(/1X,"Move limit=",5F12.8)'), (DML(I),I=1,N)
      WRITE (*,'(1X,"DYNAMICQ OPTIMIZATION ALGORITHM")')
      WRITE (*,'(1X,"-------------------------------")')
      WRITE (*,'(/1X,A))') 'Iter Function value  ? XNORM     RFD'
      WRITE (*,'(1X,A)') '--------------------------------------------'
      WRITE (11,'(1X,"DYNAMICQ OPTIMIZATION ALGORITHM")')
      WRITE (11,'(1X,"-------------------------------")')
      WRITE (11,'(/1X,A)')
     *    'Iter Function value      ? XNORM     RFD'
      WRITE (11,'(5(1X,A,I2,A,8X))') ('X(',I,')',I=1,N)

C      Initialize outer loop counter
      KLOOP=0

C      Arbitrary large values to prevent premature termination
      F_LOW=1.D6
      RFD=1.D6
      RELXNORM=1.D6

      DO I=1,(NP+NL+NU+N)
            C_A(I)=0.D0
      END DO

C     Specify initial approximation curvatures
      ACURV=0.D0
      DO I=1,NP
            BCURV(I)=0.d0
      END DO
      DO I=1,NQ
            CCURV(I)=0.D0
      END DO

C*****START OF OUTER OPTIMIZATION LOOP*********************************C

      DO WHILE (KLOOP.LE.KLOOPMAX)

C*****APPROXIMATE FUNCTIONS********************************************C

C      Determine function values
            CALL FCH(N,NP,NQ,X,F,C,H,KLOOP,0)

C      Calculate relative step size
            IF (KLOOP.GT.0) THEN
                  DELXNORM=0.D0
                  XNORM=0.D0
                  DO I=1,N
                        DELXNORM=DELXNORM+(X_H(KLOOP-1,I)-X(I))**2
                        XNORM=XNORM+X(I)**2
                  END DO
                  DELXNORM=DSQRT(DELXNORM)
                  XNORM=DSQRT(XNORM)
                  RELXNORM=DELXNORM/(1.D0+XNORM)
            END IF

C      Determine lowest feasible function value so far
C            FEASLIM=1.D-6
            IF (KLOOP.GT.0) THEN
                  FEASIBLE=.TRUE.
                  DO I=1,NP
                        IF (C(I).GT.FEASLIM) THEN
                              FEASIBLE=.FALSE.
```

```
                        END IF
                  END DO
                  DO I=1,NQ
                        IF (DABS(H(I)).GT.FEASLIM) THEN
                              FEASIBLE=.FALSE.
                        END IF
                  END DO
                  DO I=1,NL
                        IF (C_A(I+NP).GT.FEASLIM) THEN
                              FEASIBLE=.FALSE.
                        END IF
                  END DO
                  DO I=1,NU
                        IF (C_A(I+NP+NL).GT.FEASLIM) THEN
                              FEASIBLE=.FALSE.
                        END IF
                  END DO
            END IF

C       Calculate relative function difference
            IF (F_LOW.NE.1.D6.AND.FEASIBLE) THEN
                  RFD=DABS(F-F_LOW)/(1+DABS(F))
            END IF


            IF (FEASIBLE.AND.F.LT.F_LOW) THEN
                  F_LOW=F
            END IF

C       Store function values
            DO I=1,N
                  X_H(KLOOP,I)=X(I)
            END DO
            F_H(KLOOP)=F
            DO I=1,NP
                  C_H(KLOOP,I)=C(I)
            END DO
            DO I=1,NL
                  C_H(KLOOP,NP+I)=C_A(NP+I)
            END DO
            DO I=1,NU
                  C_H(KLOOP,NP+NL+I)=C_A(NP+NL+I)
            END DO
            DO I=1,N
                  C_H(KLOOP,NP+NL+NU+I)=C_A(NP+NL+NU+I)
            END DO
!              C_H(KLOOP,NP+NL+NU+1)=C_A(NP+NL+NU+1)
            DO I=1,NQ
                  H_H(KLOOP,I)=H(I)
            END DO

C       Determine gradients
            CALL GRADFCH(N,X,NP,NQ,GF,GC,GH,F,C,H,DELTX,KLOOP)

C       Calculate curvatures
            IF (KLOOP.GE.1) THEN
                  DELXNORM=0.D0
                  DO I=1,N
                        DELX(I)=X_H(KLOOP-1,I)-X_H(KLOOP,I)
                        DELXNORM=DELXNORM+DELX(I)**2
                  END DO

C       Dot product of GF and DELX
                  DP=0.D0
                  DO I=1,N
                        DP=DP+GF(I)*DELX(I)
                  END DO
```

University of Pretoria etd – De Kock, D J (2005)

```
                  ACURV=2.*(F_H(KLOOP-1)-F_H(KLOOP)-DP)
     *                  /DELXNORM

                  DO J=1,NP
C      Extract relevant vector from GC
                        DO I=1,N
                              GC_V(I)=GC(J,I)
                        END DO

C      Dot product of GC_V and DELX
                  DP=0.D0
                  DO I=1,N
                        DP=DP+GC_V(I)*DELX(I)
                  END DO

C      Calculate corresponding cuvature BCURV(J)
                  BCURV(J)=2.*(C_H(KLOOP-1,J)-C_H(KLOOP,J)
     *                  -DP)/DELXNORM
                  END DO

                  DO J=1,NQ
C      Extract relevant vector from GC
                        DO I=1,N
                              GH_V(I)=GH(J,I)
                        END DO

C      Dot product of GH_V and DELX
                  DP=0.D0
                  DO I=1,N
                        DP=DP+GH_V(I)*DELX(I)
                  END DO

C      Calculate corresponding curvature CCURV(J)
                  CCURV(J)=2.*(H_H(KLOOP-1,J)-H_H(KLOOP,J)
     *                  -DP)/DELXNORM
                  END DO
            END IF

C*****RECORD PARAMETERS FOR THE ITERATION*****************************C

C      Write approximation constants to Approx.out
            WRITE (10,'(/1X,A,I3)') 'Iteration ',KLOOP
            WRITE (10,'(1X,A)')    '--------------'
            WRITE (10,'(1X,A)') 'X='
            WRITE (10,'(1X,5(F12.8))') (X(I), I=1,N)
            WRITE (10,'(/1X,A,D15.8)') 'F=',F
            DO I=1,NP
                  WRITE (10,'(1X,A,I2,A,D15.8)') 'C(',I,')=',C(I)
            END DO
            DO I=1,NQ
                  WRITE (10,'(1X,A,I2,A,D15.8)') 'H(',I,')=',H(I)
            END DO

            WRITE (10,'(/1X,A,D15.8)') 'Acurv=',ACURV
            DO I=1,NP
                 WRITE (10,'(1X,A,I2,A,D15.8)') 'Bcurv(',I,')=',BCURV(I)
            END DO
            DO I=1,NQ
                 WRITE (10,'(1X,A,I2,A,D15.8)') 'Ccurv(',I,')=',CCURV(I)
            END DO

C      Write solution to file
            IF (KLOOP.EQ.0) THEN
                  WRITE (11,'(1X,I4,1X,D19.12,1X,L1)')
     *            KLOOP,F,FEASIBLE
                  WRITE (11,'(5(1X,D13.6))') (X(I),I=1,N)
            ELSE
                  IF (RFD.NE.1.D6) THEN
```

```
      WRITE (11,'(1X,I4,1X,D19.12,1X,L1,1X,D9.3,1X,D9.3)')
     *                  KLOOP,F,FEASIBLE,RELXNORM,RFD
                  WRITE (11,'(5(1X,D13.6))') (X(I),I=1,N)
                  ELSE
      WRITE (11,'(1X,I4,1X,D19.12,1X,L1,1X,D9.3,10X)')
     *                  KLOOP,F,FEASIBLE,RELXNORM
                  WRITE (11,'(5(1X,D13.6))') (X(I),I=1,N)
                  END IF
            END IF

C      Write solution to screen
            IF (KLOOP.EQ.0) THEN
                  WRITE (*,'(1X,I4,1X,D14.7,1X,L1)') KLOOP,F,FEASIBLE
            ELSE
                  IF (RFD.NE.1.D6.AND.FEASIBLE) THEN
                      WRITE (*,'(1X,I4,1X,D15.8,1X,L1,1X,D9.3,1X,D9.3)')
     *                  KLOOP,F,FEASIBLE,RELXNORM,RFD
                  ELSE
                        WRITE (*,'(1X,I4,1X,D15.8,1X,L1,1X,D9.3)')
     *                  KLOOP,F,FEASIBLE,RELXNORM
                  END IF
            END IF

C      Exit do loop here on final iteration
          IF (KLOOP.EQ.KLOOPMAX.OR.RFD.LT.FTOL.OR.RELXNORM.LT.XTOL) THEN
                  IF (KLOOP.EQ.KLOOPMAX) THEN
                  WRITE(*,'(1X,A)') 'Terminated on max number of steps'
                  WRITE(11,'(1X,A)') 'Terminated on max number of steps'
                  END IF
                  IF (RFD.LT.FTOL) THEN
                       WRITE(*,'(1X,A)') 'Terminated on function value'
                       WRITE(11,'(1X,A)') 'Terminated on function value'
                  END IF
                  IF (RELXNORM.LT.XTOL) THEN
                        WRITE(*,'(1X,A)') 'Terminated on step size'
                        WRITE(11,'(1X,A)') 'Terminated on step size'
                  END IF
                  WRITE(*,'(1X,A)') ' '
                  WRITE(11,'(1X,A)') ' '
                  EXIT
            END IF

C*****SOLVE THE APPROXIMATED SUBPROBLEM********************************C

            CALL LFOPC77(N,X,NP,NQ,F,C,H,GF,GC,GH,ACURV,BCURV,CCURV,DML,
     *            F_A,C_A,H_A,NL,NU,NLV,NUV,V_LOWER,V_UPPER,xtol,KLOOP)

C      Record solution to approximated problem

            WRITE (10,'(/A)') 'Solution of approximated problem:'
            WRITE (10,'(/1X,A)') 'X='
            WRITE (10,'(1X,5(F12.8))') (X(I), I=1,N)
            WRITE (10,'(/1X,A,D15.8)') 'F_A=',F_A
            DO I=1,NP+NL+NU+N
                  WRITE (10,'(1X,A,I2,A,D15.8)') 'C_A(',I,')=',C_A(I)
            END DO
            DO I=1,NQ
                  WRITE (10,'(1X,A,I2,A,D15.8)') 'H_A(',I,')=',H_A(I)
            END DO

C      Increment outer loop counter
            KLOOP=KLOOP+1
      END DO

C      Write final constraint values to file

      WRITE (11,'(/1X,A)') 'Final function value:'
      WRITE (11,'(1X,A,D19.12)') 'F=',F
```

```
      IF (NP.GT.0) THEN
            WRITE (11,'(/1X,A)') 'Final inequality constraint
     * function values:'
            DO I=1,NP
                  WRITE (11,'(1X,A,I2,A,D15.8)') 'C(',I,')=',C(I)
            END DO
      END IF
      IF (NQ.GT.0) THEN
            WRITE (11,'(/1X,A)') 'Final
     * equality constraint function values:'
            DO I=1,NQ
                  WRITE (11,'(1X,A,I2,A,D15.8)') 'H(',I,')=',H(I)
            END DO
      END IF
      IF (NL.GT.0) THEN
            WRITE (11,'(/1X,A)') 'Final side (lower)
     * constraint function values:'
            DO I=1,NL
                   WRITE (11,'(1X,A,I2,A,D15.8)') 'C(X(',NLV(I),'))='
     *            ,C_A(NP+I)
            END DO
      END IF
      IF (NU.GT.0) THEN
            WRITE (11,'(/1X,A)') 'Final side (upper)
     * constraint function values:'
            DO I=1,NU
                  WRITE (11,'(1X,A,I2,A,D15.8)') 'C(X(',NUV(I),'))='
     *            ,C_A(NP+NL+I)
            END DO
      END IF

C       Write final X values to screen (DDK 23/11/2000)
       WRITE (*,'(1/X,A)') 'Final design variables values:'
       DO I=1,N
             WRITE (*,'(1X,A,I2,A,D15.8)') 'X(',I,')=',X(I)
       ENDDO
       WRITE (*,*)

C      Write final constraint values to screen
      IF (PRNCONST) THEN
            PAUSE 'Constraint values follow: Press enter to continue'
            IF (NP.GT.0) THEN
                  WRITE (*,'(/1X,A)') 'Final inequality constraint
     * function values:'
                  DO I=1,NP
                        WRITE (*,'(1X,A,I2,A,D15.8)') 'C(',I,')=',C(I)
                  END DO
            END IF
            IF (NQ.GT.0) THEN
                  WRITE (*,'(/1X,A)') 'Final
     * equality constraint function values:'
                  DO I=1,NQ
                        WRITE (*,'(1X,A,I2,A,D15.8)') 'H(',I,')=',H(I)
                  END DO
            END IF
            IF (NL.GT.0) THEN
                  WRITE (*,'(/1X,A)') 'Final side (lower)
     * constraint function values:'
                  DO I=1,NL
                       WRITE (*,'(1X,A,I2,A,D15.8)') 'C(X(',NLV(I),'))='
     *            ,C_A(NP+I)
                  END DO
            END IF
            IF (NU.GT.0) THEN
                  WRITE (*,'(/1X,A)') 'Final side (upper)
     * constraint function values:'
                  DO I=1,NU
                       WRITE (*,'(1X,A,I2,A,D15.8)') 'C(X(',NUV(I),'))='
```

```
     *            ,C_A(NP+NL+I)
                  END DO
            END IF
      END IF

      RETURN
      END SUBROUTINE DYNQ77
```

gradfch77.for

```
C**********************************************************************C
C     USER SPECIFIED SUBROUTINE                                        C
C                                                                      C
      SUBROUTINE GRADFCH(N,X,NP,NQ,GF,GC,GH,F_X,C_X,H_X,DELTX,KLOOP)
C                                                                      C
C     COMPUTE THE GRADIENT VECTORS OF THE OBJECTIVE FUNCTION F,        C
C     INEQUALITY CONSTRAINTS C, AND EQUALITY CONSTRAINTS H             C
C     W.R.T. THE VARIABLES X(I):                                       C
C              GF(I),I=1,N                                             C
C              GC(J,I), J=1,NP I=1,N                                   C
C              GH(J,I), J=1,NQ I=1,N                                   C
C                                                                      C
C**********************************************************************C
      IMPLICIT REAL*8 (A-H,O-Z),INTEGER(I-N)
      DIMENSION X(N),GF(N),GC(NP,N),GH(NQ,N)
      DIMENSION DX(N),C_X(NP),H_X(NQ),C_D(NP),H_D(NQ)
      DIMENSION DELTX(N)

C     Determine gradients by finite difference

!     DELTX=1.D-5 ! Finite difference interval

!       CALL FCH(N,NP,NQ,X,F_X,C_X,H_X)
      DO I=1,N
             DO J=1,N
                    IF (I.EQ.J) THEN
                           DX(J)=X(J)+DELTX(J)
                    ELSE
                           DX(J)=X(J)
                    END IF
             END DO
             CALL FCH(N,NP,NQ,DX,F_D,C_D,H_D,KLOOP,I)
             GF(I)=(F_D-F_X)/DELTX(I)
             DO J=1,NP
                    GC(J,I)=(C_D(J)-C_X(J))/DELTX(J)
             END DO
             DO J=1,NQ
                    GH(J,I)=(H_D(J)-H_X(J))/DELTX(J)
             END DO
      END DO

       RETURN
      END SUBROUTINE GRADFCH
```

fch77.for

```
C**********************************************************************C
C     USER SPECIFIED SUBROUTINE                                        C
C                                                                      C
      SUBROUTINE FCH(N,NP,NQ,X,F,C,H,KLOOP,NLOOP)
```

```
C                                                                      C
C     COMPUTE OBJECTIVE FUNCTION : F                                   C
C             INEQUALITY CONSTRAINT FUNCTIONS : C                      C
C             EQUALITY CONSTRAINT FUNCTIONS : H                        C
C                                                                      C
C                                                                      C
C**********************************************************************C
      IMPLICIT REAL*8 (A-H,O-Z),INTEGER(I-N)
      DIMENSION X(N),C(NP),H(NQ)
      integer ios,node
      real time,maktke,tke

C Fluent Simulation (running on lnx86 DDK - 2002)
C Call Pre-processor
      OPEN (33,FILE='parameterset.jou')
      DO i=1,N
             WRITE (33,'(A,I1,A,D20.8)') '$x',i,' = ',X(i)
       END DO
      CLOSE(33)
C Call Gambit
       CALL SYSTEM('gambit_script')
! For windows use the file command below
!      CALL SYSTEM('gambit_script.bat')

C Call Fluent
      CALL SYSTEM('fluent_script')
! For windows use the file command below
!      CALL SYSTEM('fluent_script.bat')
!      Example of what should be in script: fluent 3d -g -wait -i input.jou

C Call Post
      maxtke = 0.
       ! Extract maximum tke from file
       open(99,file='tke.out')
       read(99,*)
       read(99,*)
       read(99,*)
       read(99,*)
       ios=0
       READ (99, *,iostat=ios) time,tke
       DO WHILE (ios.eq.0)
!         Check maximum
          if (tke.gt.maxtke) maxtke=tke
          READ (99, *,iostat=ios) time,tke
       END DO
       close(99)
      F = maxtke

! Move case, data etc. to new file name
      OPEN (33,FILE='move_script')
       WRITE (33,'(A, I2.2, A, I2.2, A)')
     #  'mv groot_impact.cas.gz groot_impact',KLOOP,'_',NLOOP,'.cas.gz'
       WRITE (33,'(A,I2.2,A,I2.2,A)')
     #  'mv groot_impact.dat.gz groot_impact',KLOOP,'_',NLOOP,'.dat.gz'
       WRITE (33,'(A,I2.2,A,I2.2,A)')
     #  'mv tke.out tke',KLOOP,'_',NLOOP,'.out'
      CLOSE(33)
      CALL SYSTEM('chmod u+x move_script')
      CALL SYSTEM('./move_script')
      CALL SYSTEM('rm move_script')

! Constraint
      C(1) = 200. - x(2) + x(3)

       RETURN
      END SUBROUTINE FCH
```

D

APPENDIX D: GAMBIT JOURNAL FILES FOR IMPACT PAD CREATION AND MESH GENERATION

This appendix contains the two Gambit journal files that were used to automatically create the impact pad and mesh the geometry used in Case study 3 of Chapter 4. The first file (gambitsetup.jou) creates the impact pad according to the supplied design variables. The design variables are defined as variables at the start of the journal (e.g. $l, $h1 etc). The second file (groot_opt.jou) imports the impact pad created by the previous file into an existing model of the tundish. The journal file then moves the impact pad to the correct location and does the necessary operations on it, after which the geometry is meshed.

gambitsetup.jou

```
/ Impact pad creation
/ DdK 19/2/2002
/ DdK 13/2/2002 - Added dam at outlet
/ Define paramaters
$w1 = 650.
$w2 = 250.
$l  = <<l>>
$h1 = <<h1>>
$h2 = <<h2>>
$d1 = <<d1>>
```

```
$d2 = <<d2>>
$d3 = <<d3>>
$p1 = <<p1>>
$h3 = <<h3>>

volume create width $l depth $h1 height $w1 offset ($l/2.) ($h1/2.) ($w1/2) brick
vertex cmove "vertex.5" multiple 1 offset 50 0 0
vertex cmove "vertex.5" multiple 1 offset 50 (-$h2) 0
vertex cmove "vertex.1" multiple 1 offset 0 130 0
edge create straight "vertex.9" "vertex.10" "vertex.11"
vertex copy "vertex.10"
vertex align "vertex.12" translation "vertex.5" "vertex.6"
coordinate create cartesian vertices "vertex.10" "vertex.11" "vertex.12"
vertex create onedge "edge.14" uparameter 0.5
vertex move "vertex.13" offset 0 0 $d2
edge create nurbs "vertex.10" "vertex.13" "vertex.11" interpolate
edge delete "edge.14" lowertopology
face create translate "edge.13" "edge.15" onedge "edge.8"
coordinate activate "c_sys.1"
vertex cmove "vertex.11" multiple 1 offset 0 0 -50
vertex cmove "vertex.16" multiple 1 offset 0 0 50
vertex cmove "vertex.14" multiple 1 offset 0 0 (($w1-$w2)/2.))
vertex cmove "vertex.10" multiple 1 offset 0 0 (-($w1-$w2)/2.))
edge create straight "vertex.20" "vertex.17"
edge create straight "vertex.19" "vertex.18"
vertex create onedge "edge.22" uparameter 0.5
vertex create onedge "edge.23" uparameter 0.5
coordinate create cartesian vertices "vertex.20" "vertex.17" "vertex.12"
vertex cmove "vertex.21" multiple 1 offset 0 ($d1) 0
coordinate create cartesian vertices "vertex.19" "vertex.18" "vertex.10"
vertex cmove "vertex.22" multiple 1 offset 0 ($d1) 0
coordinate activate "c_sys.1"
edge create nurbs "vertex.19" "vertex.24" "vertex.18" interpolate
edge create nurbs "vertex.20" "vertex.23" "vertex.17" interpolate
face create translate "edge.24" "edge.25" vector 0 200 0
face split "face.8" connected face "face.9"
face split "face.11" connected face "face.10"
```

```
edge delete "edge.23" "edge.22" lowertopology
face delete "face.12" lowertopology
volume create translate "face.8" "face.11" vector 0 -50 0
edge split "edge.6" vertex "vertex.15" connected
edge split "edge.3" vertex "vertex.16" connected
edge split "edge.54" vertex "vertex.36" connected
face create wireframe "edge.9" "edge.6" "edge.18" "edge.39" "edge.43" "edge.55" "edge.12" real
volume create translate "face.22" onedge "edge.44" reverse
volume copy "volume.4"
volume align "volume.5" translation "vertex.42" "vertex.7"
volume create translate "face.4" vector 0 55 0
volume create translate "face.3" vector 50 0 0
volume delete "volume.1" lowertopology
face delete "face.7" lowertopology
vertex create onedge "edge.97" uparameter 0.5
vertex move "vertex.70" offset 0 $d3 0
vertex create edgeints "edge.65" "edge.97" real
vertex create edgeints "edge.97" "edge.85" real
edge create nurbs "vertex.72" "vertex.70" "vertex.71" interpolate
edge create straight "vertex.72" "vertex.71"
face create wireframe "edge.107" "edge.108" real
volume create translate "face.50" onedge "edge.98" reverse
face connect real
edge connect real
vertex connect real
volume unite volumes "volume.2" "volume.3" "volume.4" "volume.5" "volume.6" "volume.7" "volume.8"

edge round "edge.113" radius1 20 radius2 0
edge round "edge.116" radius1 0 radius2 20
edge round "edge.124" "edge.114" "edge.52" "edge.123" "edge.128" "edge.118" "edge.127" "edge.115" "edge.34" "edge.117" "edge.16"
radius1 20
volume blend "volume.2"

/New wall
volume create width 50 depth $h3 height 1000 offset -25 ($h3/2) 400 brick
volume align "volume.3" translation "vertex.8" "vertex.2"
volume move "volume.3" offset (-$p1) 0 0
```

```
export acis "../../impact.sat" ascii sequencing version "6.0"
abort
```

groot_opt.jou

```
/ Import, move + subtract created Impact Pad
import acis "impact.sat" ascii
vertex create onedge "edge.1392" uparameter 0.5
vertex create onedge "edge.1584" uparameter 0.5
volume align "volume.259" "volume.260" translation "vertex.1267" "vertex.1266" rotation1 \
  "vertex.1260" "vertex.1133" rotation2 "vertex.1253" "vertex.1202"

volume create translate "face.886" onedge "edge.1392" reverse
volume create translate "face.893" onedge "edge.1392"
volume create translate "face.896" onedge "edge.1572" reverse
volume create translate "face.906" "face.914" onedge "edge.1630"
volume subtract "volume.256" volumes "volume.259" "volume.261" "volume.262" \
  "volume.265" "volume.263" "volume.264"
volume subtract "volume.255" volumes "volume.260"

/ Merge complex faces
face merge "face.867" "face.882" "face.951" "face.911"
face merge "face.885" "face.952" "face.903" "face.953"
edge merge "edge.1528" "edge.1515" forced
edge merge "edge.1519" "edge.1608" forced
edge merge "edge.1600" "edge.1583" forced
edge merge "edge.1574" "edge.1620" forced
edge merge "edge.1710" "edge.1714" forced
edge merge "edge.1695" "edge.1690" forced
edge merge "edge.1719" "edge.1720" forced
edge merge "edge.1700" "edge.1683" forced
```

```
edge merge "edge.1527" "edge.1543" "edge.1551" forced
edge merge "edge.1586" "edge.1597" "edge.1592" forced

face merge "face.887" "face.926"
face merge "face.897" "face.927"
face merge "face.877" "face.925"
edge merge "edge.1590" "edge.1636" forced
edge merge "edge.1567" "edge.1635" forced

face merge "face.881" "face.869" "face.871" "face.891"
face merge "face.899" "face.901" "face.873" "face.890"
face merge "face.884" "v_face.971" "face.889" "face.894" "v_face.964" \
  "face.898" "face.880" "face.892" "v_face.972" "face.875"
edge merge "edge.1545" "edge.1523" forced
edge merge "edge.1580" "edge.1579" forced
edge merge "edge.1536" "edge.1525" "edge.1538" forced
edge merge "edge.1559" "edge.1549" "edge.1571" forced
edge merge "edge.1729" "edge.1730" forced
edge merge "edge.1727" "edge.1726" forced

face merge "face.507" "face.511"
face merge "face.504" "face.506"
edge merge "edge.946" "edge.941" forced
edge merge "edge.944" "edge.952" forced

/Start Meshing
volume delete "volume.247" lowertopology
blayer create first 8 growth 1.2 total 42.944 rows 4 transition 1 trows 0
blayer attach "b_layer.1" volume "volume.248" face "face.516"
/At inlet
edge mesh "edge.961" "edge.844" successive ratio1 1 intervals 28
edge mesh "edge.842" successive ratio1 1 intervals 20
edge mesh "edge.1450" successive ratio1 1 intervals 15
face mesh "face.438" map pintervals 10 size 30
edge mesh "edge.1450" successive ratio1 1 intervals 15
volume mesh "volume.252" cooper source "face.831" "face.437" "face.522" "face.838" size 30
/Impact pad
```

```
edge mesh "v_edge.1761" "v_edge.1759" "v_edge.1758" "v_edge.1760" \
  "v_edge.1756" "edge.1569" "v_edge.1757" "edge.1728" "v_edge.1763" \
  "v_edge.1762" successive ratio1 1 size 12
edge mesh "edge.1725" "edge.1682" "v_edge.1753" "edge.1723" "edge.1731" \
  "edge.1686" "v_edge.1751" "edge.1732" "v_edge.1748" "v_edge.1746" \
  "v_edge.1755" "v_edge.1754" "v_edge.1747" "v_edge.1749" "edge.1711" \
  "edge.1718" successive ratio1 1 size 20
edge mesh "edge.1712" "edge.1717" successive ratio1 1 intervals 3
face mesh "face.855" map size 30
face mesh "face.854" map size 30
volume mesh "v_volume.259" tetrahedral size 30
/Inlet region
face mesh "face.412" map pintervals 12 size 20
face mesh "face.411" map pintervals 10 size 20
volume mesh "volume.220" cooper source "face.437" "face.410" size 20
/Middle
edge mesh "edge.1738" "edge.1737" successive ratio1 1 intervals 3
volume mesh "volume.255" submap size 40
/Exit area
edge mesh "edge.1426" successive ratio1 1 intervals 26
volume mesh "volume.248" cooper source "face.816" "face.818" size 30
edge mesh "edge.1514" "edge.1513" "edge.939" "edge.921" "edge.1512" successive ratio1 1 size 5
edge modify "v_edge.1764" backward
edge mesh "v_edge.1765" "v_edge.1764" firstlength ratio1 5 size 7
edge mesh "edge.945" "edge.947" successive ratio1 1 size 7
volume mesh "v_volume.260" tetrahedral size 30
edge mesh "edge.918" successive ratio1 1 intervals 30
volume mesh "volume.257" tetrahedral size 7
face mesh "face.486" map pintervals 30 size 15
volume mesh "volume.216" cooper source "face.487" "face.485" size 15

/BC
physics modify "wall_slag" btype face "face.805" "face.831" "face.816"
physics modify "wall_side" btype face "face.806" "face.813" "face.810" \
  "face.809" "face.808" "face.807" "face.812" "face.804" "face.958" \
  "v_face.966" "v_face.968" "v_face.976" "face.833" "face.843" "face.834"
physics modify "wall_bot" btype face "face.950" "face.959" "face.960" \
```

```
  "face.814"
physics modify "wall_stop_ann" btype face "face.508" "face.503" "face.489" \
  "face.516" "v_face.977" "v_face.978"
physics modify "tundish_inlet_wall" btype face "face.411" "face.438" \
  "face.412"
physics modify "SEN_inlet_wall" btype face "face.486" "face.502"

/Export mesh
export fluent5 "tundish.msh"
abort
```

APPENDIX E: FLUENT JOURNAL FILE FOR SIMULATION OF TUNDISH WITH IMPACT PAD AND EXTRACTION OF RESULTS

This appendix contains the Fluent journal file (fluent_setup.jou) that was used to automatically set-up the Fluent model after the mesh had been generated automatically in Gambit. The journal also automatically runs the simulation and writes out the appropriate files for the data extraction needed for the optimisation algorithm.

fluent_setup.jou

```
;; Read full mesh
f/sbo no yes no no
f/s-t fluent.trn
f/r-c tundish.msh
grid/scale 0.001 0.001 0.001
;; Define models
def/mod/energy yes no no no yes
;; def/mod/visc/ke-rng yes
def/mod/visc/ke yes
/def/mat/cc air liquid-steel yes boussinesq 6975 yes constant 817.3 yes constant 31 yes constant 6.4e-3  no no no yes 0.0001107 no yes
/def/oc/grav yes 0 0 -9.81
/def/oc/ot 1773.15
```

```
;; BCs
/def/bc/mfi mass_inlet yes 33.25 no 1823.15 no 0 no yes yes yes no no no yes 5 0.084
/def/bc/wall wall_side 0 no 0 no no no -9000 no no no 0 0.5
/def/bc/wall wall_bot 0 no 0 no no no -6000 no no no 0 0.5
/def/bc/wall wall_slag 0 no 0 no no no -18000 no no yes shear-bc-spec-shear 0 0.5 no 0 no 0 no 0
;; Monitors
;/solve/monitors/residual/plot yes
;; Initialise
solve/init/cd/mfi mass_inlet
solve/init/sd z 0
solve/init/if
;; Solution Controls
solve/set/ur/p 0.2
solve/set/ur/mom 0.5
solve/set/ur/t 0.7
solve/set/ur/k 0.6
solve/set/ur/e 0.6
solve/set/ur/d 0.7
solve/set/ur/bf 0.7
;; Steady State
;; 1st Order
solve/iter 300
adapt/aty+ 50 200 0 0 yes
solve/iter 50
adapt/miir no mass-imbalance -0.0001 0.0001
adapt/atr 0 0 0 yes
solve/iter 200
;; 2nd Order
solve/set/ds/p 12
solve/set/ds/mom 1
solve/set/ds/t 1
solve/set/ds/k 1
solve/set/ds/e 1
solve/iter 200
adapt/miir no mass-imbalance -0.0001 0.0001
adapt/atr 0 0 0 yes
solve/iter 500
solve/set/ur/mom 0.4
solve/iter 50
```

```
;; Post
/plot/plot yes tke.out yes no no tke yes 1 0 0 wall_slag ()
;; Set-up particles
;                                                          random_eddy num_tries                                diam
/def/inj/ci injection-0 no yes surface no mass_inlet no yes    no        100       0.15 no no no 0 0 -0.7692084 6.9e-6 1823.15 0
/def/mat/cc anthracite alumina yes 3200 yes constant 1680 yes 31 yes
/def/inj/ci injection-1 no yes surface no mass_inlet no yes    no        100       0.15 no no no 0 0 -0.7692084 1.65e-5 1823.15 0
/def/inj/ci injection-2 no yes surface no mass_inlet no yes    no        100       0.15 no no no 0 0 -0.7692084 2.30e-5 1823.15 0
/def/inj/ci injection-3 no yes surface no mass_inlet no yes    no        100       0.15 no no no 0 0 -0.7692084 3.21e-5 1823.15 0
/def/inj/ci injection-4 no yes surface no mass_inlet no yes    no        100       0.15 no no no 0 0 -0.7692084 4.52e-5 1823.15 0
/def/inj/ci injection-5 no yes surface no mass_inlet no yes    no        100       0.15 no no no 0 0 -0.7692084 6.39e-5 1823.15 0
/def/inj/ci injection-6 no yes surface no mass_inlet no yes    no        100       0.15 no no no 0 0 -0.7692084 9.03e-5 1823.15 0
/def/inj/ci injection-7 no yes surface no mass_inlet no yes    no        100       0.15 no no no 0 0 -0.7692084 0.000137 1823.15 0
;                                     #steps
/define/models/dpm/tracking/max-steps 100000
; Trap on slag layer
/define/bc/wall wall_slag 0 no 0 no no no -18000 no no no 0 0.5 yes trap no 0 no 0 no 0
; Run particles
/file/stop-t
/file/s-t part.trn
/report/dpm-sample-report injection-7 injection-6 injection-5 injection-4 () outflow () () no
/file/stop-t
;; Write case and data
file/wcd
groot_impact.cas.gz

exit
```

University of Pretoria etd – De Kock, D J (2005)

F

APPENDIX F: FORTRAN PROGRAM TO EXTRACT DATA

This appendix gives the two Fortran77 programs that were used to extract the data that were written to file during the Fluent simulation. The first program (post.for) extracts the maximum turbulent kinetic energy on the free surface of the molten steel. The second program (post_part.for) extracts the number of particles that has been trapped on the free surface of the molten steel. These programs are used in conjunction with LS-OPT to automate the optimisation procedure.

post.for

```
      program ls_opt_post
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! This program takes data from Fluent simulation    !
! calculate the response and the give to LSOPT      !
!                                                   !
! DDK 1/2002                                        !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

      real time,maxk,k
      integer ios
      character*255 line

      maxk = 0.
! Extract maximum tke from file
      open(99,file='tke.out')
      read(99,*)
      read(99,*)
      read(99,*)
      read(99,*)
      ios=0
      READ (99, *,iostat=ios) time,k
      DO WHILE (ios.eq.0)
!        Check maximum
         if (k.gt.maxk) maxk=k
         READ (99, *,iostat=ios) time,k
      END DO
      close(99)

! Echo to standard output
      write(*,*) maxk

      end program ls_opt_post
```

post_part.for

```
      program ls_opt_post
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! This programs takes data from Fluent simulation   !
! calculate the response and give to LSOPT          !
!                                                   !
! The specific response is the number of particles  !
! trapped                                           !
!                                                   !
! DDK 7/2002                                        !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

      real perc_trapped
      integer tracked,escaped
      integer ios
      character*255 line,line2,line3

! Extract particles trapped from FLUENT
      open(99,file='part.trn')
      read(99,'(A150)') line
      do while (line.ne.'DPM Iteration ....')
             read(99,'(A150)') line
      enddo
      read(99,'(A150)') line
      read(99,'(A16, I4, A11, I4, A150)') line,
     +              tracked,line2,escaped,line3
      close(99)
      perc_trapped=(tracked-escaped)/(1.0*tracked)

      write(*,*) perc_trapped

      end program ls_opt_post
```