The University of Manchester

SCHOOL OF MECHANICAL, AEROSPACE AND CIVIL ENGINEERING
FACULTY OF ENGINEERING AND PHYSICAL SCIENCES

PO BOX 88
MANCHESTER, UK
M60 1QD

TEL: +44(0)161 306 9200
FAX: +44(0)161 306 3755

FLUID DYNAMICS, POWER GENERATION
AND ENVIRONMENT DEPARTMENT

SINGLE PHASE THERMAL-HYDRAULICS GROUP
6, QUAI WATIER
F-78401 CHATOU CEDEX

TEL: 33 1 30 87 75 40
FAX: 33 1 30 87 79 16

Code_Saturne version 2.0.0 tutorial

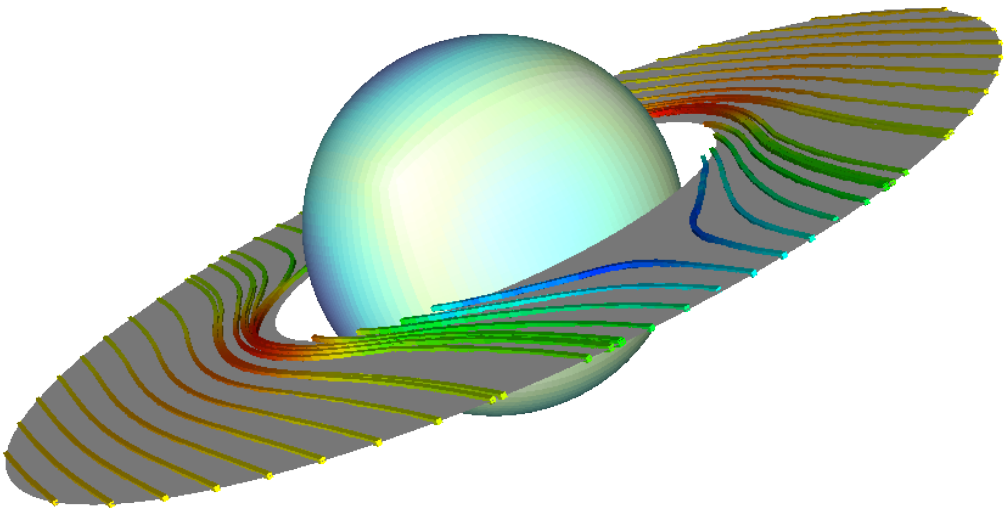

TABLE OF CONTENTS

Part I

2D DIFFUSER

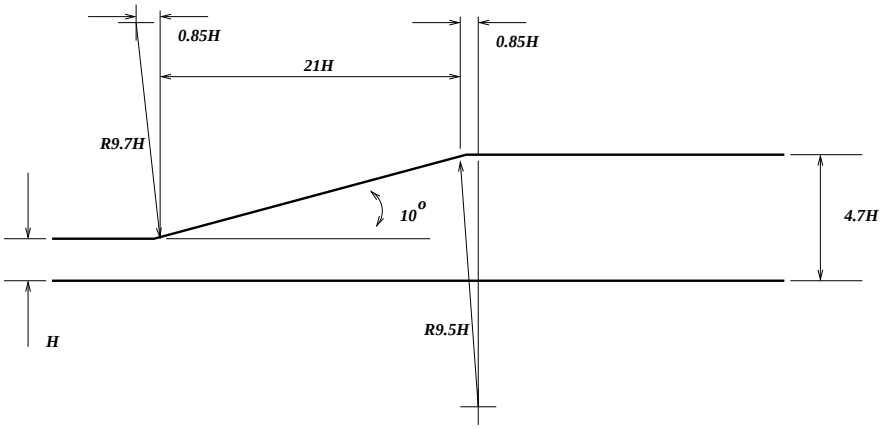

Figure I.1: Asymmetric plane diffuser. Geometry description.

1 Introduction

The aim of this tutorial is to give an insight of *Code_Saturne* to new users by solving the asymmetric plane diffuser case. The case was done experimentally by Buice and Eaton [1]. This flow in a planar asymmetric diffuser is a challenging test case since its adverse pressure gradient makes the flow separate from an inclined surface. The diffuser has the geometry described in figure I.1

The Reynolds number based on the inlet height H and the bulk velocity is $Re = 18000$. This case was first treated experimentally by Obi et al. [5] and then later by Buice and Eaton [1]. The same diffuser was a case for the 8th ERCOFTAC/IAHR Workshop [2]. The case was also studied by Kaltenbach et al. [3] using Large Eddy Simulation. The case is interesting from the modelling point of view because of the well defined boundary conditions (fully developed channel flow), the separation that occurs smoothly due to the low angle of the inclined wall and the recovery region after the reattachment. Determining the separation correctly depends on the models capabilities to predict a correct turbulent time scale along the accelerated wall. In order to mimic correctly the reattachment and recovery, the model will need to predict the correct magnitude of the shear stress.

The inlet conditions for this case are defined as a fully developed channel flow, therefore is recommended to calculate a channel flow with the same turbulence model that is to be used in the computation. In this tutorial, a mesh with periodic boundary conditions will be used to get the channel flow results and then use them as an inlet condition for the diffuser.

2 Requirements

Before using *Code_Saturne* you will need to update your PATH with the location of the binary files (note that this depends on the local installation). Export your PATH by typing:

```
[bash:$] export PATH=$PATH:/opt/cs-2.0-rc2/bin
```

3 Use of *Code_Saturne*

The necessary steps to run a case in *Code_Saturne* are:

1. Prepare the case.
2. Create the mesh (if it doesn't exists) and copy it to the correct location.
3. Modify the user subroutines.
4. Modify the *runcase* launch script.
5. Launch the calculation.
6. Visualise the results.

This document provides an example for two cases, a periodic channel flow and asymmetric plane diffuser. The meshes can be found here:

http://cfd.mace.manchester.ac.uk/Main/DiffuserTutorial.

Additional software required to complete this tutorial include `xmgrace` and `ParaView` in order to be able to visualise the results.

4 Channel Flow

The aim of this section is to compute the flow in a channel (i.e. flow between two infinite parallel plates) at a Reynolds number of 18000 based on the bulk velocity and the channel height. The profiles obtained here are going to be used as inlet boundary conditions for the asymmetric plane diffuser case.

4.1 Prepare the case

Code_Saturne requires an specific directory structure which can be created by typing:

```
[bash:$] code_saturne create --study CHANNEL SST --nogui
```

This will create an directory structure as seen in figure I.2.

The option `--study` creates a study called `CHANNEL` with a first case called `SST` since the turbulence model to be used is the SST [4]. The details of the structure are:

- SST: This is the case file, a study can have several cases, for example different turbulence models, different boundary conditions etc.

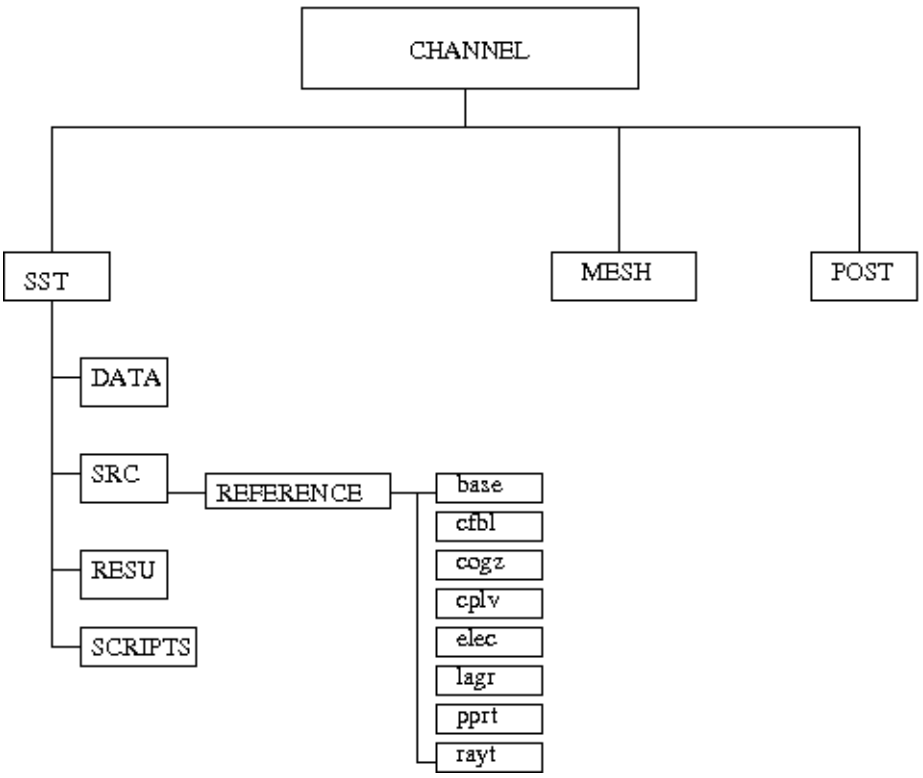

Figure I.2: File structure created by *Code_Saturne*

  - DATA: This is where the restarting files need to be copied, it will be also used in this tutorial to store the file with the prescribed inlet conditions.
  - SRC: This is where the user subroutines are going to saved. During the compilation stage, the code will re-compile all the subroutines that are in this directory.

    In the `REFERENCE` directory all the user-defined subroutines are stored. For a specific case you will only need a few of them depending on the changes that you want to implement on the code. The `base` directory contains the basic flow solver subroutines, the other directories contain the subroutines needed by other modules such as the Lagrangian, the combustion or the electric modules. For more information on the other modules please see the "Practical user guide to *Code_Saturne* " (you can get it by typing `code_saturne info -g user`).
  - RESU: Directory where the results will be exported once the calculation is finished.
  - SCRIPTS: Here is were the launching scripts are copied. You will need to run the executable file *runcase* to start the calculation.

- MESH: The program will read the mesh from this directory. Some of the mesh formats that can be read are I-DEAS, CGNS, Gambit Neutral and CD-Adapco files.

- POST: This is an empty directory designed to contain post-processing macros (xmgrace, exper-

imental etc.)

4.2 Copying and checking the mesh

Once you have created the directory structure, you will need to go to the MESH directory by typing:

```
[bash:$] cd CHANNEL/MESH
```

The mesh for the channel is a 1D mesh, it has only one cell on the x direction and 50 in the vertical (y) direction with an inlet height of 1 (see figure I.3). Since the idea is to use the results as an inlet boundary conditions, the mesh has the same locations for the cell centres as the diffuser mesh in order to avoid interpolation. Download the mesh and save it in the MESH directory. If the mesh is

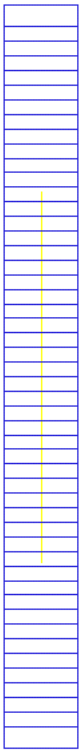

Figure I.3: Mesh for the inlet channel

compressed (*.*gz*) you can used the following command to unzip

```
[bash:$] gzip -d *.gz
```

To check that the mesh is ready for use in *Code_Saturne*, the *preprocessor* can be used. Type:

```
[bash:$] code_saturne check_mesh  -m channeldiff50x1.unv
```

The output should give you the coordinates of the geometry of the mesh and the number of cells and faces including the boundary faces along with some other information. Verify that there are no errors on the output window. This commands runs the preprocessor and parts of the kernel that deal with the geometrical entities created during a calculations. The output can be checked in the *check_mesh.ensight/* directory using *ParaView*

It is important that the boundary faces can be distinguishable in order to apply the correct boundary conditions. In this case the output from the $cs_preprocess command will show the 'colours' that are defined in the mesh$

```
Number of vertices                    :      204
Number of faces                       :      302
                              Colour 5   :        2
                              Colour 9   :       50
                              Colour 1   :       50
                              Colour 4   :      200
Number of cells                       :       50
                              Colour 0   :       50
```

These 'colours' correspond to the following boundary conditions:

Colour	5	wall
Colour	9	inlet
Colour	1	outlet
Colour	4	Symmetry

These "colours" will be used later to impose periodicity between the inlet and the outlet.

4.3 User defined subroutines

Code_Saturne is written in FORTRAN90 but was written originally in FORTRAN 77. There are several user subroutines that can be used to modify certain parameters in a calculation. All the user subroutines are in the directory `SRC/REFERENCE`. When you want to use any user defined subroutine, it is necessary to copy it to the `SRC/` directory in order to compile it. For any case, if you are not using the graphical interface, there are two subroutines which always need to be modified, these are are `usini1.f90` and `usclim.f90` located in the directory `REFERENCE/base`. The `usini1.f90` subroutine is where all the parameters of the calculation have to be defined, these include numerical parameters such as, turbulence models, convection schemes etc. and physical parameters such as viscosity, density, gravity etc. The `usclim.f90` subroutine is where the boundary conditions are prescribed.

For this case you will need to copy the following subroutines into the `SRC/` directory and modify them:

```
usclim.f90  usini1.f90  usproj.f90  ustsns.f90
```

The `usproj.f90` subroutine is used to write output files and the `ustsns.f90` subroutine is used to change the source terms in the momentum equation. In the next sections a description of the required modifications is presented (you can use kwrite, vi, gvim, emacs, gedit, kate or any other text editor). Go to the `SRC` directory and copy the subroutines by typing:

```
[bash:$] cd  ../SST/SRC
[bash:$] cp REFERENCE/base/usclim.f90 .
[bash:$] cp REFERENCE/base/usini1.f90 .
[bash:$] cp REFERENCE/base/usproj.f90 .
[bash:$] cp REFERENCE/base/ustsns.f90 .
```

usini1.f90

Most of the default parameters do not need to be changed for this calculation. For more information on the parameters listed in this subroutine see "Practical user guide to *Code_Saturne*". The parameters that need to be changed are:

- Turbulence model: Set the value of `iturb(iphas)` to 60 for $k-\omega$ SST

```
! --- Turbulence (for each phase)
!       0...Laminar
!      10...Mixing length
!      20...k-epsilon
!      21...k-epsilon (linear production)
!      30...Rij-epsilon, (standard LRR)
!      31...Rij-epsilon (SSG)
!      40...LES (Smagorinsky)
!      41...LES (Dynamic)
!      42...LES (WALE)
!      50...v2f (phi-model)
!      60...k-omega SST
!  For 10, contact the development team before use

iphas = 1
iturb(iphas) = 60
```

- Number of time steps: Set the value of `ntmabs` to 2000

```
! --- Duration
!       ntmabs = absolute number of the last time step required
!         if we have already run 10 time steps and want to
!         run 10 more, ntmabs must be set to 10 + 10 = 20

ntmabs = 2000
```

- Reference time step: set the value of `DTREF` to 0.1

```
! --- Reference time step
!     The example given below is probably not adapted to your case.

dtref  = 0.1d0
```

- Density and viscosity: Set the value of `ro0` to 1 and the value of `visc0` to $1/18000$

```
iphas = 1

ro0(iphas)    = 1.d0
visc0(iphas) = 1.d0/18000.d0
cp0(iphas)    = 1219.d0
```

- Reference velocity: Set the value of `uref(iphas)` to 1.

- Captors: You can set captors at any point of the domain to obtain the convergence history. The value of `ncapt` is the number of captors that you want to use and the table `xyzcap` is where the locations of the captors are stored.

```
! --- Number of monitoring points (probes) and their positions
!     (limited to ncaptm=100)

ncapt  = 5
xyzcap(1,1) = 5.0d0
xyzcap(2,1) = 0.0d0
xyzcap(3,1) = 0.01d0

xyzcap(1,2) = 5.0d0
xyzcap(2,2) = 0.01d0
xyzcap(3,2) = 0.01d0

xyzcap(1,3) = 5.0d0
xyzcap(2,3) = 0.1d0
xyzcap(3,3) = 0.01d0
```

```
xyzcap(1,4) = 5.0d0
xyzcap(2,4) = 0.5d0
xyzcap(3,4) = 0.01d0

xyzcap(1,5) = 5.0d0
xyzcap(2,5) = 0.4d0
xyzcap(3,5) = 0.01d0
```

usclim.f90

This is where the boundary conditions are defined. Since the periodicity is going to be defined in the launching script file, you only have to define the wall and symmetry boundary conditions. This file has examples of different types of boundary conditions including inlets calculated from the reference velocity and turbulent intensity. The function *getfbr* is used to obtain the faces that match a selection criteria given in the arguments of the functions. These criteria can be geometrical ($x < 0$) or by properties or names (inlet, outlet etc.) given by the software used to create the mesh. For more information on how to set non-standard boundary conditions see the "Practical user guide to *Code_Saturne*". The only test that you have to leave in this subroutine are those on walls and symmetry faces, which as explained before, correspond to colours 5 and 4. Edit the file and delete the lines that are not necessary, at the end your file should only have the following calls to *getfbr*:

```
!===============================================================
! 1.  Initialization
!===============================================================

idebia = idbia0
idebra = idbra0

d2s3 = 2.d0/3.d0
!===============================================================
! 2.  Assign boundary conditions to boundary faces here

!     One may use selection criteria to filter boundary case subsets
!       Loop on faces from a subset
!         Set the boundary condition for each face
!===============================================================

! --- Prescribe at boundary faces of color 5 a wall for all phases
call getfbr('5', nlelt, lstelt)
!==========
do ilelt = 1, nlelt
  ifac = lstelt(ilelt)
```

```
  ! Wall: zero flow (zero flux for pressure)
  !       friction for velocities (+ turbulent variables)
  !       zero flux for scalars
  do iphas = 1, nphas
    itypfb(ifac,iphas)   = iparoi
  enddo
enddo
!
! --- Prescribe at boundary faces of color 4 a symmetry
call getfbr('4', nlelt, lstelt)
!==========
do ilelt = 1, nlelt
  ifac = lstelt(ilelt)
  ! Symmetries
  do iphas = 1, nphas
    itypfb(ifac,iphas)   = isymet
  enddo
enddo
!----
! Formats
!----
!----
! End
!----
return
end subroutine
```

usproj.f90

This is the file where a user-defined output file can be defined. It contains several examples on how to do things like calculate budgets, extract profiles and do things in parallel. It is very useful since it is called at each time step. Here we are going to use it to write the values of all variables in the domain into a file at the last time step. Note that *ntcabs* is the current time step and *ntmabs* is the absolute number of time steps. The turbulence model is set via *iturb* and the variables we are interested in are: the y coordinate (*xyzcen(2,iel)*), the stream wise velocity (*rtp(iel,iu(1))*), the turbulent kinetic energy k (*rtp(iel,ik(1))*) and the rate of dissipation ω (*rtp(iel,iomg(1))*). You will need to delete the examples and leave the following:

```
!===============================================================
! 1.  Initialization
!===============================================================
! Memory management
```

```
idebia = idbia0
idebra = idbra0
if(ntmabs.eq.ntcabs)then
  open(file="inlet.dat", unit=impusr(1))
  do iel =1, ncel
   if(iturb(1).eq.60)then
      write(impusr(1), '(4e17.9)') xyzcen(2,iel)          &
     ,rtp(iel,iu(1)), rtp(iel,ik(1))          &
     ,rtp(iel,iomg(1))
   endif
  enddo
  close(impusr(1))
endif
return
end subroutine
```

The file written is called "inlet.dat" and we use the special array *impusr()* which contains pointers available for the user files. Later you will have to modify the launching script to be able to recover this file from the temporary location.

ustsns.f90

This subroutine is used to change the source terms in the momentum equation. Since we have a periodic flow, it is necessary to add a pressure gradient in order to drive the flow. The pressure gradient will be calculated from the mass flux and will be updated every time step.

In this subroutine, the implicit additional source terms will be stored in `crvimp` and the explicit ones in `crvexp`. You will need to use only the explicit part so the file should look like:

```
!===============================================================
integer          iifac, ifac
double precision ubulk, ubulka, flux, surf, surfx, surfy, surfz, xrel
common           / glob / ubulk, ckp

!===============================================================
! 1. Initialization
!===============================================================
idebia = idbia0
idebra = idbra0
! ----------------------------------------------
iphas=1
if (ivar.eq.iu(1)) then
! Initialisation
  if(ntcabs.eq.ntpabs-1) then
```

```
    ckp = 0.d0
    ubulk  = uref(1)
  endif
  ubulka = ubulk
  flux = 0.d0
  surf = 0.d0
! relaxation factor
  xrel = 0.9d0
! loop on all faces and find the ones that are at x=0
! compute the total surface and the total mass flux
  do ifac = 1, nfac
    if(abs(cdgfac(1,ifac)-0.d0).lt.1.d-6) then
      surfx = surfac(1,ifac)
      surfy = surfac(2,ifac)
      surfz = surfac(3,ifac)
      surf  = surf + sqrt(surfx**2 + surfy**2 + surfz**2)
      iifac = iifac + 1
      flux  = flux + propfa(ifac,ipprof(ifluma(iu(1))))
    endif
  enddo
! for parallel calculations, sum on all processors
  if(irangp.ge.0)then
    call parsom(flux)
    call parsom(surf)
    call parcpt(iifac)
  endif
! obtain bulk velocity by dividing flux by area
  ubulk = abs(flux) / surf
! correct the source term with the difference between
! the reference and teh actual.
  ckp = xrel*ckp + (1.d0-xrel)*(uref(1) - ubulk)
  write(nfecra, *) "Total␣number␣of␣faces␣at␣inlet:", iifac
  write(nfecra, *) "Total␣surface␣at␣inlet:", surf
  write(nfecra, *) "Bulk␣velocity␣at␣time␣step␣N:", ubulk
  write(nfecra, *) "Bulk␣velocity␣at␣time␣step␣N-1:", ubulka
  write(nfecra, *) "Reference␣bulk␣velocity:", uref(1)
  write(nfecra, *) "Relaxation␣factor:", xrel
  write(nfecra, *) "Total␣correction␣term:",ckp
!
  do iel = 1, ncel
     crvexp(iel) =   volume(iel)*ckp
  enddo
!
```

```
endif
return
end subroutine
```

Note the definition of the new local variables at the beginning. We only need to activate this for the stream wise component of the velocity vector, that is why there is a test *if(ivar.eq.iu(1))*.

4.4 Scripts and launching

The launching scripts are in the `SCRIPTS` directory, the one you will use is `runcase`. This script creates the necessary links and directories in order to run a case, compiles the subroutines and writes the results. You will need to carry out the following modifications:

- Change the name of the mesh in the variable `MESH` in the line 137 and add the command for the periodicity `COMMAND_PERIO`:

```
MESH=channeldiff50x1.unv                              !  CHANGE THIS
COMMAND_REORIENT=
COMMAND_JOIN=
COMMAND_CWF=
COMMAND_PERIO="--perio␣--trans␣0.1␣0␣0␣--color␣1␣9" ! CHANGE THIS
THERMOCHEMISTRY_DATA=
METEO_DATA=
```

  The periodicity command tell the code that there is one translational periodicity, gives the vector of that translation and the colour of the faces that comprise that periodicity.

- Recover the results: By default, *Code_Saturne* will save the results in an Ensight output format but you will have to add the file `inlet.dat`. In the line 158 of the script, add this file name to the list of files that are being copied into the `RES_USER` directory

```
#
USER_INPUT_FILES=""
USER_OUTPUT_FILES="inlet.dat"                 ! CHANGE THIS
#
```

Once you have completed all these modifications you can run the case by typing `./runcase`. This will create a directory called `tmp_Saturne` in the case directory where all the temporary files are stored. If there is a compilation error, a file called `compil.log` is copied to the `RESU` directory. Once the calculation has started successfully you can go to the `tmp_Saturne` in your home directory[1] and check a file called `listing` where the output of every time step is written.

[1]The location of the `tmp_Saturne` can be changed by editing the variable CS_TMP_PREFIX in the *runcase* file.

4.5 Results

Once the calculation is finished you can check the results in the `RESU` directory. All the relevant files have the date suffixed to the file name so it is easy to distinguish them from different calculations (i.e. listing.MMDDHHMM). The `CHR.*` directory contains the results in a Ensight format, they can be seen using 'Paraview'. The `HIST.*` directory contains history files for all the captors in the domain (a file for each variable). If you want to check the convergence history go to that directory and type:

```
[bash:$] code_saturne plot_probes probes_XXX.dat
```

where `probes_XXX.dat` is the file for the variable that you want to check (for example "probes_VelocityX.dat"). The `listing.*` file contains the general information about the calculation and some lines for each time step including then maximum and minimum value of each variable and the Courant number.

The folder `RES_USER.*` contains the user's files including `inlet.dat.*` which contains the velocity, energy and dissipation rate profiles needed for the the inlet boundary condition of the diffuser case. If you want to look at the profiles, go to the `RES_USER.*` directory and type:

```
[bash:$] xmgrace -nxy inlet.dat &
```

The `RESTART.*` directory has the necessary data for restarting the calculation, if you want to do this, you should copy this directory in the `DATA` directory, changing the name to `RESTART` and set the value of *isuite* to 1 in the `usini1.f90` file.

5 Diffuser Case

After completing the channel flow calculation the necessary inlet boundary conditions can be used with the diffuser. The steps to run the calculation are very similar to the ones described for the channel flow and a simplified explanation is given below:

5.1 Prepare the case

Create a new study called "DIFFUSER" with a case called "SST" by typing:

```
[bash:$] cd
[bash:$] code_saturne create --study DIFFUSER SST --nogui
```

This should give the same structure that the previous study. In the `DATA` directory, copy the resulting file from the channel flow calculation without the date. For example, if you created the study in the same directory as the channel flow study you can type:

```
[bash:$] cp CHANNEL/SST/RESU/RES_USER*/inlet.dat.* DIFFUSER/SST/DATA/inlet.dat
```

5.2 Copying the mesh

Copy the diffuser mesh *DiffIC292x50y25.neu* into the `MESH` directory from the website given at the beginning of this document. An image of the diffuser region is shown in figure I.4.

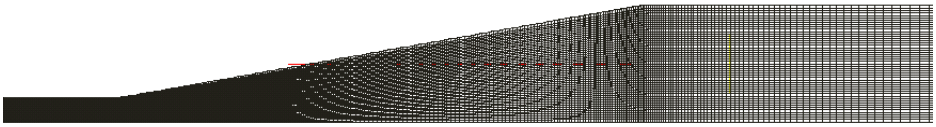

Figure I.4: Diffuser mesh

The mesh has 292x50 cells and is intended for high Reynolds models since it does not resolve the viscous sublayer.

5.3 User subroutines

The subroutines needed for this case are:

`usclim.f90  usini1.f90 usproj.f90`

usini1.f90

The parameters that need to be changed are:

- Turbulence model: Set `iturb(iphas) = 60` for the same model as in the previous computation (SST).
- Number of time steps: Set `ntmabs = 2000`
- Time step: Set `dtref = 1`
- Set the converge criteria to 10^{-3} for all the variables by changing the value of the `epsilo` array. You can do this by adding a line into the loop on all variables:

```
ntmabs = 2000
! --- Reference time step
!     The example given below is probably not adapted to your case.
dtref  = 1.d0
!
do ii = 1, nvar
  epsilo(ii) = 1.d-3
enddo
```

- Density and Viscosity: Set the value of `rO0 = 1.d0` and `viscl0 = 1.d0/18000.d0`

- History captors: Since the domain in this case is much bigger than the previous case you should put more captors to be sure that the calculation has converged. Use these locations:

```
ncapt  = 10
xyzcap(1,1) = 0.0d0
xyzcap(2,1) = 0.05d0
xyzcap(3,1) = 0.01d0

xyzcap(1,2) = 0.30d0
xyzcap(2,2) = 0.00d0
xyzcap(3,2) = 0.01d0

xyzcap(1,3) = 0.30d0
xyzcap(2,3) = 0.08d0
xyzcap(3,3) = 0.01d0

xyzcap(1,4) = 6.0d0
xyzcap(2,4) = 0.05d0
xyzcap(3,4) = 0.01d0

xyzcap(1,5) = 6.0d0
xyzcap(2,5) = 0.5d0
xyzcap(3,5) = 0.01d0

xyzcap(1,6) = 10.0d0
xyzcap(2,6) = 0.00d0
xyzcap(3,6) = 0.01d0

xyzcap(1,7) = 10.d0
xyzcap(2,7) = 4.50d0
xyzcap(3,7) = 0.01d0

xyzcap(1,8) = 20.0d0
xyzcap(2,8) = 4.7d0
xyzcap(3,8) = 0.01d0

xyzcap(1,9) = 40.0d0
xyzcap(2,9) = 4.7d0
xyzcap(3,9) = 0.01d0

xyzcap(1,10)= 60.0d0
xyzcap(2,10)= 4.70d0
```

```
xyzcap(3,10)= 0.01d0
```

- User arrays: Since you are going to read the inlet profiles from a file, it is necessary to allocate memory for an array that can handle these values. Set the value of `nrtuse` to `4*50` since it is necessary to store four variables (position, velocity, kinetic energy and dissipation rate) over 50 faces at the inlet.

usclim.f90

This file has to be modified to impose the appropriate boundary conditions, a copy of the file can be obtained from the web since it is too large to be included in this guide. The subroutine imposes standard boundary conditions for the wall, symmetry and outlet faces and it opens the "inlet.dat" file, reads the data and store them in the user array *rtuser*. Afterwards it loops on the 50 entries from the file and finds the nearest inlet face in the y direction. After, it sets the corresponding values for the other variables.

5.4 usproj.f90

This subroutine will be used to create several data files for comparison with the experimental data available. A copy of a file with all the locations of the experiments is available from web page. This subroutine will create two files (*buicstr.cpp* and *buicinc.cpp*) with pressure and friction coefficients, one for each wall. Indeed 21 files with stream wise velocity, kinetic energy and dissipation rate, at different x locations, will be written in order to match the experimental data (*buicsat*.uvw*)

5.5 Scripts and launching

As in the previous case you will need to modify the `runcase` script. Change the name of the mesh in the variable `MESH` to `DiffIC292x50y25.neu` and link the inlet data file by adding the line with the name of the file. In order to copy the files with the profiles add the following to the output files variable:

```
#
USER_INPUT_FILES="inlet.dat"
USER_OUTPUT_FILES="bui*␣ads*"
#
```

Once you have done all the modifications, you can launch the calculation by executing the runcase file. On a Intel at 2.13MHz, the calculation takes about 45 minutes. You can check the progress of the calculation by going into the `tmp\_Saturne` directory. Check the message on the terminal in order to see where the temporary execution directory is located. Go into the temporary directory

and you will find that for each run there is an execution directory named STUDY.CASE.DATE. This avoids overwriting any previous results, but that also means you have to clean it manually once the calculation is finished.

5.6 Results

Once the calculation has finished the results can be viewed in the `RESU*` directory. The convergence can be checked in the `HIST*` directory and the Ensight results can be viewed using Paraview in the `CHR*` directory.

An example of the comparisons that can be made is the friction coefficient along the inclined wall which can show where separation occurs. By using xmgrace the experimental and the computational results can be easily compared. The experimental results are available on the web at the following address:
`http://cfd.mace.manchester.ac.uk/Saturne/DiffuserTutorial` Type

```
[bash:$] xmgrace &
```

and then go to $>$ *Data* $>$ *Import* $>$ *ASCII* and then select the file `g_cf%.y00` which has the experimental friction coefficient along the inclined wall. Then import the file `buicinc.cpp` and select the option `Block Data` with the columns 1 and 3 (see figure I.5)

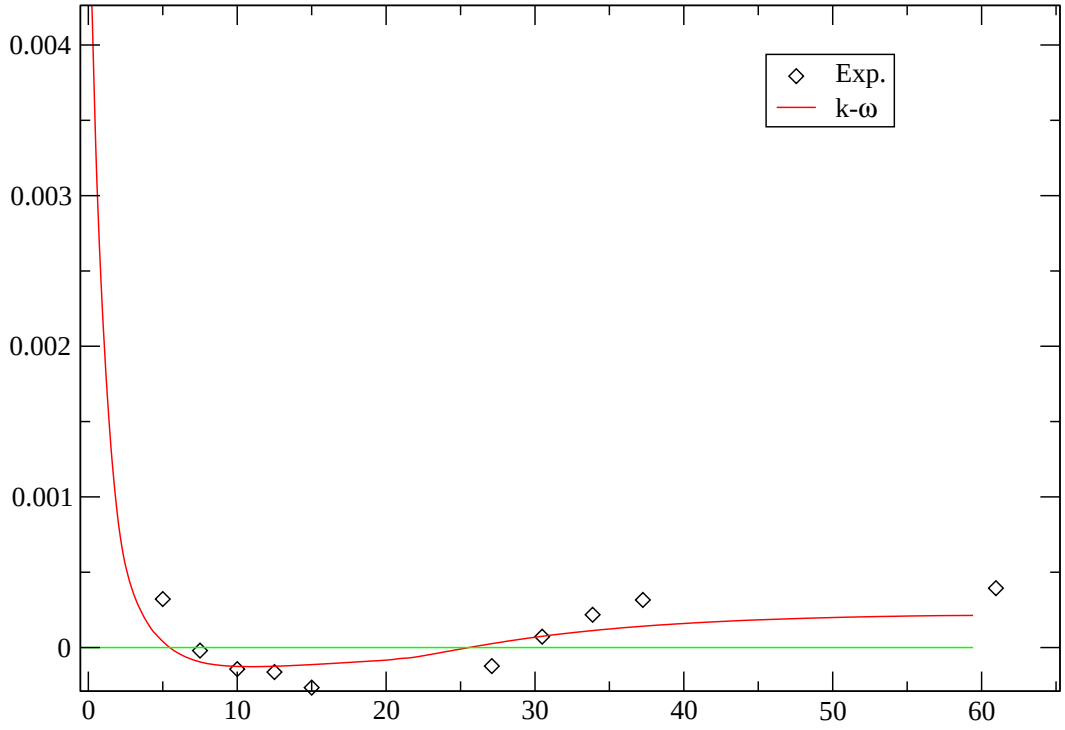

Figure I.5: Skin friction coefficient along the inclined wall

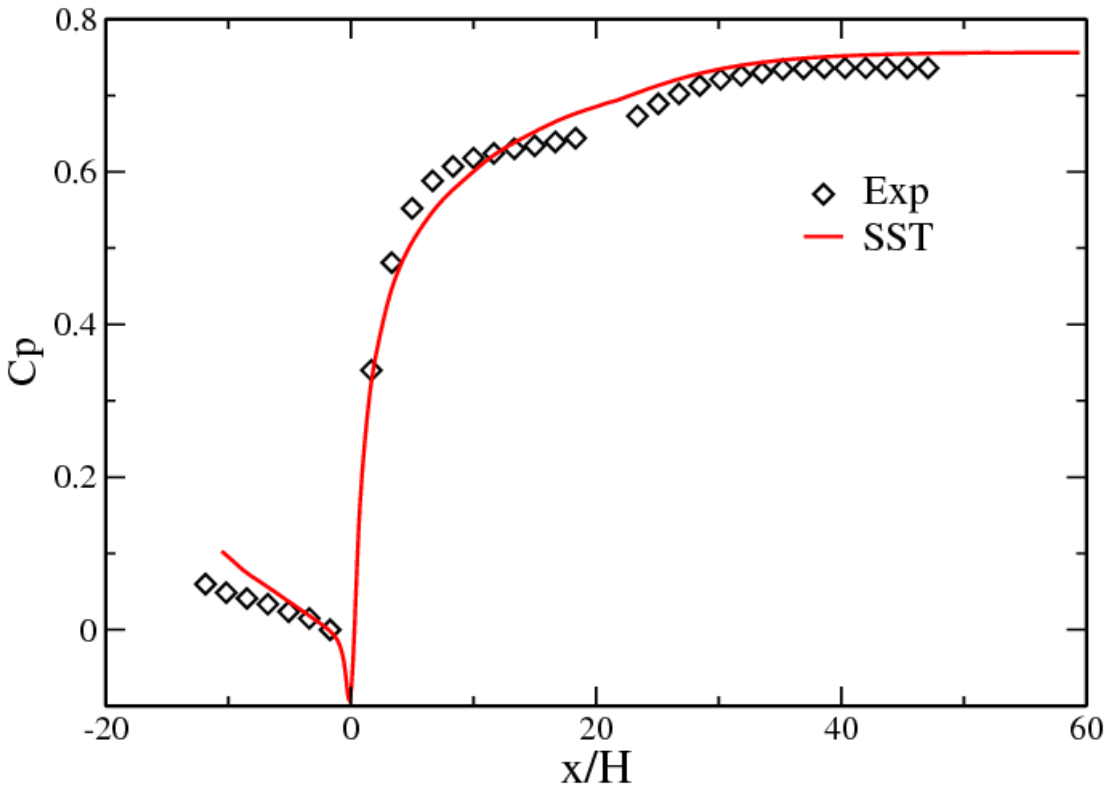

Figure I.6: Pressure coefficient along the inclined wall

Similar comparison can be done with the straight wall and the pressure coefficients (figure I.6). The velocity profiles are also available for comparison. For more information on the experimental data see: http://tmdb.ws.tn.tudelft.nl/workshop8/case8_2/case8_2.html

6 References

[1] C.U. Buice and J.K. Eaton. Experimental investigation of flow through an asymmetric plane diffuser. Technical Report TSD-107, Department of mechanical engineering, Stanford University, 1997.

[2] A. Hellsten and P. Rautaheimo, editors. *8th ERCOFTAC/IAHR/COST Workshop on refined turbulence modelling*. ERCOFTAC/IAHR/COST, 1999.

[3] H.J Kaltenbach, M. Fatica, R. Mittal, T.S. Lund, and P. Moin. Study of flow in a planar asymmetric diffuser using Large Eddy Simulation. *Journal of Fluid Mechanics*, pages 151–185, 1999.

[4] F. R. Menter. Zonal two-equation $k-\omega$ turbulence models for aerodynamic flows. In *24th AIAA Fluid Dynamics Conference, Orlando FL, Paper 93-2906*, 1993.

[5] S. Obi, K. Aoki, and S. Masuda. Experimental and computational study of turbulent separating

flow in and asymmetric plane diffuser. In F. Durst, N. Kasagi, B.E. Launder, F.W. Schmidt, K. Suzuki, and J.H. Whitelaw, editors, *Proceedings 9th Symposium of turbulent shear flow*, Kyoto, Japan, August 1993.