Simulation of a backward facing step using *hydra-th* 2015

G. Cartland Glover

13th November 2015

Abstract

Simulations of a backward facing step, where a Poiseuille flow profile is applied to the inlet condition have been performed. The code *hydra-th* is used. Three Reynolds numbers, Re, were used (300, 600 and 800), which are in the laminar regime. The percentage error in comparison to simulated values obtained from relevant literature was less than -4 % at $Re =$300, ±5 % at $Re =$600 and ±10 % at $Re =$800.

1 Introduction

To verify the open source fluid dynamic codes we use, simple tests cases are being developed. The cases should be quick to run and have analytical or results published in literature against which comparisons can be made. In this report, we discuss the simulation of a backward facing step, with a Poiseuille flow profile applied to the inlet condition. The locations of at the separation and reattachment points were compared with literature values reported by Barton (Barton, I. E., A numerical study of flow over a confined backward-facing step, *International Journal for Numerical Methods in Fluids* **21(8)**, 653–665, 1995).

The code used here is *hydra-th* (`https://get-hydra.lanl.gov/`), a highly parallelisable code developed at Los Alamos National Laboratory for the Consortium for the Advanced Simulation of Light Water Reactors (CASL) lead by Oak Ridge National Laboratory (`http://www.casl.gov/Hydra.shtml`). *hydra-th* is a cell-centred incompressible flow solver using a discontinuous Galerkin framework with a hybrid finite element/finite volume discretisation, which reduces to a finite volume method as the resolution is reduced. A high resolution advection algorithm can apply both implicit and explict advection, while time integration includes the backward Euler and trapezoidal methods.

2 Models

The flow domains considered were simple two-dimensional geometries. The mesh used was 1 cell thick due to solver requirements with no-slip walls applied at the upper and lower surfaces (Fig. 1). At the upstream end, an inlet condition is applied on the upper half of the surface with a no-slip wall on the lower half. An outlet condition is applied at the downstream end. The height of the domain, H, was specified as $1.0\,\mathrm{m}$ and the height of the inlet was $H/2$.

Two domains of lengths $15H$ and $30H$ were used. Only one mesh was specified for the $15H$ case, which was used as an equivalent mesh for direct comparison with Mesh 7 employed by Barton (1995).

A Poiseuille profile with a mean velocity of $1\,\mathrm{m\,s^{-1}}$ was applied to the inlet condition via the user function given in Appendix A. This was applied by the line `user velx sideset 1` in the control file found in Appendix B. The pressure at the downstream boundary was defined as $0.0\,\mathrm{kg\,m^{-1}\,s^{-2}}$ with all other surfaces undefined. The front and back surfaces of the meshes were also treated as symmetry conditions by applying a $0.0\,\mathrm{m\,s^{-1}}$ value to the z component of the velocity.

The fluid was defined in terms of the Reynolds number, Re, where the characteristic dimension was $H = 1\,\mathrm{m}$. A density of $1.0\,\mathrm{kg\,m^{-3}}$ was assumed and the viscosity was adjusted to give $Re = 300$, 600 and 800. This gave viscosities of 3.333×10^{-3}, 1.666×10^{-3} and $1.25 \times 10^{-3}\,\mathrm{kg\,m^{-1}\,s^{-1}}$.

The simulations were performed as transient calculations with an adaptive timestep based on the maximum Courant number observed in the flow domain. The initial and maximum time step sizes also control how the calculation proceeds until the maximum time is reached. These parameters are defined as 1×10^{-3} time units and 0.2 time units. A total time of up to 600.0 time units was specified to allow the convergence of the velocity components and the pressure at points near to the reattachment and separation points.

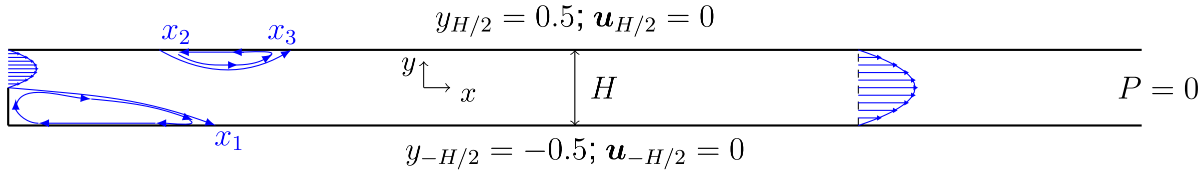

Figure 1: Geometrical configuration and boundary conditions applied to backward facing step under Poiseuille flow conditions with the rough location of the separation (x_2) and reattachment (x_1 and x_3) points. Recirculation loops are indicated by blue vectors. They are located below the inlet and upstream of x_1 as well as between x_2 and x_3. The flow profile applied at the inlet and the expected flow profile in the downstream section are also coloured blue.

Two solvers were applied to the resolve the continuity equations of mass and momentum. These were the `ppesolver` and the `momentumsolver`. A `transportsolver` was specified, but it was not used. An algebraic multigrid solver was used for the `ppesolver`, while the flexible generalised minimum residual method was used for the momentum transport equations. Note that the `momentumsolver` was preconditioned by an incomplete LU factorisation.

The flow fields were analysed with `pvpython` by plotting the data on 15 m lines that were located 1×10^{-3} m from the upper and lower surfaces (see Appendix C for the corresponding script). These data was then extracted in the form of a comma separated values file. The file contains data of the pressure, the velocity components and the components of the vorticity. A *NumPy–Python* script was prepared and used to interpolate (`numpy.interp`) the locations at which the streamwise velocity component was zero from each datafile containing the extracted data (see Appendix C). A similar analysis was performed using the z-component of the vorticity, as a similar transition between positive and negative values was also observed at the reattachment and separation points. Information on the maximum pressure and the key variables at this location was also extracted. These data are not presented here, as the maximum pressure of each dataset is downstream of the reattachment and separation points.

3 Results

Tables 1-6 in Appendix D present the separation (x_2) and reattachment points (x_1 and x_3) of the flow on the upper and lower surfaces of the backward facing step domain. Table 1 gives results obtained from Barton (1995) including the values of Gartling (1990) that were reported therein. The values of x_1, x_2 and x_3 reported by Barton (1995), were given in terms of the dimensionalisation that was used. The dimensions of the duct Barton (1995) used were $2H$ by $30H$ with an inlet height of H that was H above the lower surface. Consequently the values of x_1, x_2 and x_3 are twice the values calculated by the geometries reported here. Therefore, the corresponding values for the dimensionalisation used here are given in Table 1 in parentheses.

Eight meshes were used in the simulation of the backward facing step with resolutions between the coarsest of 41 nodes by 801 nodes and the finest of 161 by 4001. Node doubling was performed between y-component resolutions of 41 and 161 at an x-component resolution of 1001. Node doubling was also performed between 41 by 1001 and 161 by 4001. A mesh that was equivalent to Mesh 7 used by Barton (1995) with the corresponding aspect ratio was also simulated. The meshes had uniform node distributions with the exception of the two meshes, which had 41 by 801 nodes and 101 by 1803 nodes. The first 15 m of these meshes were defined as uniform meshes with 501 and 1501 nodes in the x-component direction. Beyond the 15 m section, variable node distributions from fine at 15 m to coarse at the outlet, were applied with the remaining nodes (i.e. 300 and 302 nodes).

3.1 $Re = 300$

Table 2 gives the values of x_1 at $Re = 300$. The average value of intersection point where the streamwise velocity component crosses the abscissa (from here on referred to as the intersection point) was 3.52 m downstream of the inlet. The corresponding average for the z-component of the vorticity was 3.51 m.

The percentage error relative to Barton's reported value varied between -3.3 % (41 by 801) and -0.9 % (161 by 4001) for the position determined by the intersection point of the x-component velocity, while for the z-component vorticity it varied between -4.0 % (41 by 801) and -1.0 %(161 by 4001).

The effect of node doubling on the accuracy of x_1 is most significant in the y-component direction, while the effect is negligible in the x-component direction when more than 1000 nodes are considered. The position of x_1 also moves downstream as the resolution in the y-component direction is doubled. There is an exception when the resolution is changed between the 101 by 201 and the 101 by 1803. Between these cases there was ~0.25 % decrease in the error.

3.2 $Re = 600$

Tables 3-4 provide the values of x_1, x_2 and x_3 at $Re = 600$. The respective averaged values of the intersection point for the streamwise velocity component were 5.26 m, 4.40 m and 8.02 m downstream of the inlet. The corresponding averages for the z-component of the vorticity were 5.25 m, 4.44 m and 7.98 m.

The percentage error relative to Barton's reported values varied between -3.9 % (41 by 1001) and 0.02 % (101 by 201) for the position determined by the intersection point of the x-component velocity at x_1, x_2 or x_3. When examining the z-component vorticity, the locations varied between -4.3 % (41 by 801) and 0.03 % (101 by 1803). Generally, the x_2 values were further away from the values for Barton's Mesh 7 than either x_1 or x_3.

The effect of node doubling in the y-component direction on x_1 was that it's position shifted from upstream of Barton's value to downstream of it. The finer meshes had an error of less than 1 %. In the case of x_2, it is always downstream of Barton's value with no discernable improvement in the accuracy. x_3 is always upstream of the corresponding location, though it does approach Barton's value as the resolution is increased. As for $Re = 300$, the effect of the number of nodes in the x-component direction is not as significant compared with in the y-component direction.

3.3 $Re = 800$

Tables 5-6 give the values of x_1, x_2 and x_3 at $Re = 800$. The respective averaged values of the intersection point for the streamwise velocity component were 5.94 m, 4.83 m and 10.33 m downstream of the inlet. The corresponding averages for the z-component of the vorticity were 5.93 m, 4.87 m and 10.29 m.

The percentage error relative to Barton's reported value varied between -8.7 % (41 by 801) and 0.25 % (101 by 201) for the position determined by the intersection point of the x-component velocity. In comparison, the location of the intersection point for the z-component vorticity were between -9.3 % (41 by 801) and 0.2 %(161 by 4001).

At $Re = 800$, the x_3 values were further away from Barton's data than either x_1 or x_2. In fact the values for x_3 correspond better with the values of Gartling (Gartling, D. K., A test problem for outflow boundary conditions – flow over a backward-facing step, *International Journal for Numerical Methods in Fluids* **11(7)**, 953–967, 1990) reported by Barton. Note that Barton also indicated that his values were 1 unit further downstream than those reported in the literature without giving any possible explanation. The values of x_1 and x_2 reported by both Barton and Gartling are similar and the simulations performed here either approach the reported values (x_1) or cover the space between the reported values of Barton and Gartling (x_2).

The effect of node doubling on x_1 resulted in a reduction of the error and a shift from the upstream side of Barton's value to the downstream side with the most significant changes observed between the doubling between 41 and 81 nodes; however, meshes above 100 nodes in the y-component direction produced the most accurate values with errors less than 1 %. As for $Re = 600$, there is no discernable improvement in the value of x_2 at higher resolutions, but at $Re = 800$, the location of x_2 shifts downstream from the upstream side of Barton's value and beyond. The change in x_3 as the number of nodes is doubled follows the behaviour of x_1 except that the error has a greater significance, as previously mentioned due to the differences between Barton's and Gartling's value for x_3.

3.4 Differences between the velocity and vorticity intersection points

It is apparent from the data in Table 2, Table 3 and Table 5 that the difference between the intersection point for the x-component velocity and the z-component vorticity reduces as the meshes are refined. The smallest differences occurred for the meshes with more than 100 nodes in the y-component direction and 1000 nodes in the x-component direction.

The corresponding data is not presented in Table 4 and Table 6 for x_2 or x_3 for reasons of clarity, but the differeneces are between the respective intersection points are 2.5 and 3.7 times greater than the values in Table 3 for $Re = 600$, while for $Re = 800$, the values are 2.8 and 4 times greater than the values in Table 5. This suggests that even the best mesh needs further refinement in order to capture better the separation and reattachment points at the upper surface. This is was also indicated by Barton, in that he found x_2, the separation point, as the hardest point to resolve.

4 Conclusions

Simulations of flow over a backward facing step have been performed using the open source code *hydra-th*. The locations of the separation and reattachment on the upper and lower surfaces were compared with those reported by Barton (1995). A number of mesh refinements were applied, and it was found that the resolution across the flow (i.e. y-component direction) had the strongest effect on the accuracy of the simulations.

5 Nomenclature

Latin symbols

x_1	Reattachment point on the lower surface (m).
x_2	Separation point on the lower surface (m).
x_3	Reattachment point on the upper surface (m).
N_x	Number of nodes in the x-direction (-).
N_y	Number of nodes in the y-direction (-).
P	Pressure ($\mathrm{kg\,m^{-1}\,s^{-2}}$).
$\boldsymbol{u}$	Velocity vector ($\mathrm{m\,s^{-1}}$).
u_x	local x-direction velocity component ($\mathrm{m\,s^{-1}}$).
u_y	local y-direction velocity component ($\mathrm{m\,s^{-1}}$).
u_z	local z-direction velocity component ($\mathrm{m\,s^{-1}}$).
x	x-direction component (m).
y	y-direction component (m).
z	z-direction component (m).

Greek symbols

ω_z	z-component of the vorticity ($\mathrm{s^{-1}}$).

A User Velocity Condition

Listing 1: C++ Code for User Velocity condition

```
//*******************************************************************************
/*!
  \file     src/FVM/CCIncNavierStokes/UserBCs/UserVelBC.C
  \author   Yidong (Tim) Xia
  \date     Thu Jul 14 12:46:23 2011
  \brief    User defined velocity boundary condition based on sidesets
 */
//*******************************************************************************
#include <cmath>

using namespace std;

#include <CCINSUserVelocityBC.h>
#include <UnsMesh.h>
#include <BCPackage.h>

using namespace Hydra;

Real
CCINSUserVelocityBC::setUserVel(UnsMesh& /* mesh */,
                                Material& /* mat */,
                                int /* edgeId */,
                                Real* coord,
                                Real /* time */) const
/*******************************************************************************
Routine: setUserVel - set a function for velocities
Author : Yidong (Tim) Xia
*******************************************************************************/
{
  // Get the physical properties upon your need
  // Real mu = mat.Viscosity();

  // Define parameters upon your need

  // Input specific functions
  Real bcval = 0.0;
  switch(m_direction) {
  case GLOBAL_XDIR:
    // Prescribe your x-velocity
    //bcval =  PI*cos(PI*coord[0])*sin(PI*coord[1])*exp(-2.0*PI*PI*mu*time);

    // CHT test problem
    //bcval = (3*0.00134/2)*(1-pow((coord[1]/0.01),2.0));
    //bcval = (3*0.827/2)*(1-pow(((coord[1]-0.5)/0.5),2.0));

    // Backward facing step for parallel plane Poiseuille flow 2h = 1, midpoint 0.25 or 0.75
    bcval = (3.0/2.0)*(1.0-pow(((coord[1]-0.25)/0.25),2.0));
  break;
  case GLOBAL_YDIR:
    // Prescribe your y-velocity
    //bcval = -PI*sin(PI*coord[0])*cos(PI*coord[1])*exp(-2.0*PI*PI*mu*time);
    bcval = 0.0;
  break;
  case GLOBAL_ZDIR:
    // Prescribe your z-velocity
```

```
    bcval =  0.0;
    break;
  default:
    assert("Unknown user velocity boundary condition direction" == 0);
  }
  return bcval;
}

Real
CCINSUserVelocityBC::setUserAcc(UnsMesh& /* mesh */,
                                Material& /* mat */,
                                int /* edgeId */,
                                Real* /* coord */) const
/*********************************************************************************
Routine: setUserAccX - set a user-defined function for acceleration
Author : Yidong (Tim) Xia
*********************************************************************************/
{
  // Get the physical properties upon your need
  // Real mu = mat.Viscosity();

  // Define parameters upon your need
  // Real rp = -2.0*PI*PI*mu;

  // Input specific functions
  Real bcval;
  switch(m_direction) {
  case GLOBAL_XDIR:
    // Prescribe your x-acceleration
    // bcval =  PI*cos(PI*coord[0])*sin(PI*coord[1])*rp*exp(rp*time);
    bcval = 0.0;
    break;
  case GLOBAL_YDIR:
    // Prescribe your y-acceleration
    //bcval = -PI*sin(PI*coord[0])*cos(PI*coord[1])*rp*exp(rp*time);
    bcval = 0.0;
    break;
  case GLOBAL_ZDIR:
    // Prescribe your z-acceleration
    bcval =  0.0;
    break;
  default:
    assert("Unknown user acceleration boundary condition direction" == 0);
  }
  return bcval;
}
```

B Control Files

Listing 2: Control file to specify simulation settings

```
title
Re=300, no entrance region

cc_navierstokes

  nsteps   100000
  deltat   0.001
  term     600.0

  solution_method
    eps_p0 1.0e-8
  end

  time_integration
    type    fixed_cfl
    CFLinit 1.0
    CFLmax  2.5
    dtmax   0.2
    dtscale 1.025
    thetaa  0.5
    thetak  0.5
    thetaf  0.5
  end

  # Output options
  pltype exodusii
  filetype serial
  plti   100
  ttyi   10
  dump   000        # Interval to write restarts

  # Material model setup & assignment to sets
  material
    id  1
    rho 1.0
    mu  0.00333333333333   # Re=300
  end

  materialset
    id 10
    material 1
    block 1
  end

  plotvar
    elem vel
    elem volume
    elem density
    elem procid
    elem div
    elem vorticity
    node vel
    node pressure
    node vorticity
  end

```

```
histvar
  elem 30000 vel
  elem 30000 pressure
  elem 2000 vel
  elem 2000 pressure
  elem 10311 vel
  elem 10315 vel
end

# BC's
# 1 - inlet
# 2 - top/bottom
# 3 - front/back
# 4 - outlet
# 5 - wall below inlet
# 6 - inlet (also 7 and 8)

pressure
  sideset 4 -1 0.0 # Outlet
end

velocity
  velx sideset 1 -1 1.0   # Inlet
  user velx sideset 1
  vely sideset 1 -1 0.0
  velz sideset 1 -1 0.0
  velx sideset 2 -1 0.0   # top/bottom wall
  vely sideset 2 -1 0.0
  velz sideset 2 -1 0.0
  velx sideset 5 -1 0.0   # wall under inlet
  vely sideset 5 -1 0.0
  velz sideset 5 -1 0.0
  velz sideset 3 -1 0.0   # back
end

ppesolver
  type  AMG
  itmax 250
  itchk 1
  coarse_size 100
  diagnostics off
  convergence off
  eps 1.0e-8
end

momentumsolver
  type  ILUFGMRES
  itmax 50
  itchk 2
  restart 20
  diagnostics off
  convergence off
  eps   1.0e-6
end

transportsolver
  type  ILUFGMRES
  itmax 50
  itchk 2
  restart 20
```

```
    diagnostics off
    convergence off
    eps   1.0e-6
  end

end

exit
```

C Python Scripts

Listing 3: ParaView python script to extract data from plots over lines

```
import sys
infile = sys.argv[1] # first argument file name
(filename,sep,exten)=infile.rpartition('.') # strip path name and extenstion
from paraview.simple import * # import librarys
reader=OpenDataFile(infile) # load results file to reader array
# open graphics window
Show()
Render()
pd=reader.PointData  # assign point data to array pd
print pd.keys() # print variable labels and ranges
for ai in pd.values():
  print ai.GetName(),ai.GetNumberOfComponents,
  for i in xrange(ai.GetNumberOfComponents()):
    print ai.GetRange(i),
  print
outfile1 = 'base'+filename+'.csv'
outfile2 = 'top'+filename+'.csv'
cd=reader.CellData # assign cell data to array cd
print cd.keys()  # print variable labels
# plot velocity in graphics window
readerRep=GetRepresentation()
readerRep.ColorArrayName ='vel'
readerRep.LookupTable = AssignLookupTable(reader.PointData['vel'],'Cool to Warm')
Render()
currentview=GetActiveView() # assign active view to currentview
currentsources=GetActiveSource() # assign active sources to currentsources
# annotate time to graphics window
annTime=AnnotateTimeFilter(currentsources)
Show(annTime)
tsteps=currentsources.TimestepValues # assign timestep values to array tsteps
print tsteps # print tsteps
nsteps=len(tsteps) # length of array tsteps and assign to array nsteps
print nsteps, tsteps[nsteps-1] # print nsteps and last value in tsteps array
currentview.ViewTime=tsteps[nsteps-1] # move data step to last time step
print currentview.ViewTime # print current time
Render() # render updated field in graphics window
for ai in pd.values(): # print variables and ranges to check difference from previous
  print ai.GetName(),ai.GetNumberOfComponents,
  for i in xrange(ai.GetNumberOfComponents()):
    print ai.GetRange(i),
  print
npoints=5000 # number of data points
plot1 = PlotOverLine(currentsources)  # use filter plotoverline
plot1.Source.Point1=[0,-0.499,0]   # define line point 1
plot1.Source.Point2=[15,-0.499,0]  # define line point 2
plot1.Source.Resolution=npoints   # define line resolution
SaveData(outfile1,FieldAssociation="Points",Precision=8) # write data to file
plot2 = PlotOverLine(currentsources)  # use filter plotoverline
plot2.Source.Point1=[0,0.499,0]   # define line point 1
plot2.Source.Point2=[15,0.499,0]  # define line point 2
plot2.Source.Resolution=npoints  # define line resolution
SaveData(outfile2, Precision=8, FieldAssociation="Points") # write data to file
```

Listing 4: Interpolation of inflexion points

```
import sys,numpy # define libraries to use
infile = sys.argv[1] # first argument : file name
col1 = int(sys.argv[2]) # second argument : column for x-component velocity
col2 = int(sys.argv[3]) # third argument : column for the x coordinates i.e. streamwise direction
col3 = int(sys.argv[4]) # fourth argument : column for the z-component vorticity
col4 = int(sys.argv[5]) # fifth argument : column for the pressure
# strip path and file name in different parts
(prefix, sep, suffix) = infile.rpartition('.')
(direct, sep2, filename) = prefix.rpartition('/')
(nodesy, sep3, nodesx) = direct.rpartition('by')
nx=int(nodesx) # number of nodes in x-direction
ny=int(nodesy) # number of nodes in y-direction
# define output files
outfile1 = prefix + '.dat' # space separated file
outfile2 = prefix +'.int'  # assessed data file
a = numpy.loadtxt(infile,delimiter=',',skiprows=1) # load file into array a, ignoring the titles in row 1
numpy.savetxt(outfile1, a) # write space separated *.dat file for plotting in gnuplot
nrows = len(a)  # find length of a or number of rows
hnrows = nrows/2 # find midway point to enable interpolation of both inflexion points at the upper surface
# identify lower and upper surface data sets
if filename[0]=='b':
# lower surface
   mrows = 0
   x=a[mrows:nrows,col1]  # assign x-component velocity to array x from start to end of array
   y=a[mrows:nrows,col2]  # assign x-coordinates to array y from start to end of array
   z=-a[mrows:nrows,col3] # assign z-component vorticity to array y from start to end of array
   c=numpy.interp(0.0,x,y)  # interpolate the x_1 reattachment point / inflexion point of zero velocity
   d=numpy.interp(0.0,z,y)  # interpolate the x_1 reattachment point / inflexion point of zero vorticity
   diffcd=c-d  # find the difference between the rettachment points estimated by the velocity and vorticity
   p=numpy.amax(a[:,col4])  # find the maximum pressure from the profile
   parg=numpy.argmax(a[:,col4]) # find the row number of the maximum pressure
   px=a[parg,col2]  # return the x coordinate at the row number for the maximum pressure
   pu=a[parg,col1]  # return the x-component velocity at the row number for the maximum pressure
   pvortz=a[parg,col3] # return the z-component vorticity at the row number for the maximum pressure
   g=numpy.column_stack((ny,nx,c,d,diffcd,parg,px,p,pu,pvortz)) # assign assessed data to array g
   numpy.savetxt(outfile2, g,delimiter='&')  # write array g to *.int file
else:
# upper surface
   mrows = 0
   x1=-a[mrows:hnrows,col1] # assign x-component velocity to array x from start to midpoint of array
   x2=a[hnrows:nrows,col1] # assign x-component velocity to array x from midpoint to end of array
   y1=a[mrows:hnrows,col2] # assign x-coordinates to array y from start to midpoint of array
```

```
    y2=a[hnrows:nrows,col2] # assign x-coordinates to array y from midpoint to end of array
    z1=-a[mrows:hnrows,col3] # assign z-component vorticity to array y from start to midpoint of array
    z2=a[hnrows:nrows,col3] # assign z-component vorticity to array y from midpoint to end of array
    c=numpy.interp(0.0,x1,y1)  # interpolate the x_2 separation point / inflexion point of zero velocity
    d=numpy.interp(0.0,x2,y2)  # interpolate the x_2 separation point / inflexion point of zero vorticity
    e=numpy.interp(0.0,z1,y1)  # interpolate the x_3 reattachment point / inflexion point of zero velocity
    f=numpy.interp(0.0,z2,y2)  # interpolate the x_3 reattachment point / inflexion point of zero vorticity
    diffce=c-e  # find the difference between the separation points estimated by the velocity and vorticity
    diffdf=d-f # find the difference between the rettachment points estimated by the velocity and vorticity
    p=numpy.amax(a[:,col4]) # find the maximum pressure from the profile
    parg=numpy.argmax(a[:,col4]) # find the row number of the maximum pressure
    px=a[parg,col2] # return the x coordinate at the row number for the maximum pressure
    pu=a[parg,col1] # return the x-component velocity at the row number for the maximum pressure
    pvortz=a[parg,col3] # return the z-component vorticity at the row number for the maximum pressure
    h=numpy.column_stack((ny,nx,c,d,e,f,diffce,diffdf,parg,px,p,pu,pvortz)) # assign assess data to array h
    numpy.savetxt(outfile2, h,delimiter='&') # write array h to *.int file
```

Listing 5: Averaging of inflexion points

```
import sys,numpy # define librarys to use
infile = sys.argv[1] # first argument : file name
# strip path and file name in different parts
(prefix, sep, suffix) = infile.rpartition('.')
outfile1 = prefix + '.plot' # space separated file
outfile2 = prefix +'.ave' # averaged data file
a = numpy.loadtxt(infile,delimiter='&') # load file into array a, ignoring the titles in row 1
numpy.savetxt(outfile1, a) # write data to space separated file for plotting
c=numpy.mean(a,axis=0) # calculate the mean
d=numpy.std(a,axis=0)  # calculate the standard deviation
numpy.savetxt(outfile2, (c,d) ,delimiter='&') # write data to *.ave file
```

D Tables

The tables presented here compare the separation and reattachment points on the lower (x_1) and upper (x_2 & x_3) wall with those by Barton (Barton, I. E., A numerical study of flow over a confined backward-facing step, *International Journal for Numerical Methods in Fluids* **21(8)**, 653–665, 1995) using Mesh 7. The values in brackets are percentage errors in respect to the relavant reference value with the exception of Table 1, which were half of the correspond values. This includes values at $Re = 800$ of a fine mesh simulation performed by Gartling (Gartling, D. K., A test problem for outflow boundary conditions – flow over a backward-facing step, *International Journal for Numerical Methods in Fluids* **11(7)**, 953–967, 1990), which was reported by Barton (1990).

The values reported in the Tables 2-6 are half the values reported by Barton (1995) due to differences in the non-dimensionalisation of the system studied. Barton (1995) used a domain $2H$ high ($H = 1.0$), where the flow entered the domain through an inlet that was H wide and positioned H above the base. Note that the domain used by Barton was $30H$. The domains used here were H high with $H/2$ for the inlet width located $H/2$ above the base. The domains were $30H$ long with the exception of the 101 by 201 which was $15H$.

Table 1: Values of x_1, x_2 and x_3 reported by Barton (1995), where Mesh 7 was used. Values in parentheses are the half values. $*$: Values reported by Barton of Gartling's fine mesh simulation at $Re = 800$;

Re	x_1	x_2	x_3
300	7.170 (3.585)	-	-
600	10.610 (5.305)	8.570 (4.285)	16.240 (8.120)
800	12.090 (6.045)	9.540 (4.770)	22.210 (11.105)
800*	12.200 (6.100)	9.700 (4.850)	20.960 (10.480)

Table 2: Comparison of x_1 values at $Re = 300$ with percentage errors in parentheses. The reference value for the reattachment length at the base of the domain is x_1 =3.585, half of the values reported by Barton (1995). *: Mesh given in testing suite of *hydra-th*; †: Equivalent to Barton's Mesh 7;

N_y	N_x	$x_{1,u_x=0}$	$x_{1,\omega_z=0}$	$x_{1,u_x=0} - x_{1,\omega_z=0}$
$101^\dagger$	201	3.533 (-1.45)	3.522 (-1.75)	1.09×10^{-2}
101	1803	3.541 (-1.22)	3.532 (-1.48)	9.41×10^{-3}
41^*	801	3.466 (-3.31)	3.443 (-3.97)	2.39×10^{-2}
41	1001	3.466 (-3.31)	3.443 (-3.97)	2.39×10^{-2}
81	1001	3.531 (-1.51)	3.518 (-1.87)	1.29×10^{-2}
161	1001	3.553 (-0.90)	3.546 (-1.10)	7.26×10^{-3}
81	2001	3.531 (-1.49)	3.520 (-1.83)	1.19×10^{-2}
161	4001	3.554 (-0.86)	3.547 (-1.05)	6.63×10^{-3}

Table 3: Comparison of x_1 values at $Re = 600$ with percentage errors in parentheses. The reference value for the reattachment length at the base of the domain is x_1 =5.305, half the reported values. *: Mesh given in testing suite of *hydra-th*; †: Equivalent to Barton's Mesh 7;

N_y	N_x	$x_{1,u_x=0}$	$x_{1,\omega_z=0}$	$x_{1,u_x=0} - x_{1,\omega_z=0}$
$101^\dagger$	201	5.306 (0.02)	5.295 (−0.19)	1.11×10^{-2}
101	1803	5.316 (0.21)	5.306 (0.03)	9.58×10^{-3}
41^*	801	5.101 (−3.84)	5.075 (−4.33)	2.57×10^{-2}
41	1001	5.099 (−3.88)	5.074 (−4.36)	2.56×10^{-2}
81	1001	5.290 (−0.28)	5.277 (−0.53)	1.33×10^{-2}
161	1001	5.344 (0.73)	5.336 (0.59)	7.25×10^{-3}
81	2001	5.293 (−0.23)	5.281 (−0.46)	1.23×10^{-2}
161	4001	5.347 (0.79)	5.340 (0.67)	6.65×10^{-3}

Table 4: Comparison of x_2 and x_3 values at $Re = 600$ with percentage errors in parentheses. The reference values for the separation and reattachment length at the base of the domain are x_2 =4.285 and x_3 =8.120, half the reported values. *: Mesh given in testing suite of *hydra-th*; †: Equivalent to Barton's Mesh 7;

N_y	N_x	$x_{2,u_x=0}$	$x_{3,u_x=0}$	$x_{2,\omega_z=0}$	$x_{3,\omega_z=0}$
$101^\dagger$	201	4.407 (2.85)	8.045 (−0.93)	4.447 (3.77)	8.014 (−1.31)
101	1803	4.420 (3.14)	8.049 (−0.88)	4.454 (3.94)	8.022 (−1.21)
41^*	801	4.352 (1.57)	7.920 (−2.46)	4.427 (3.31)	7.855 (−3.27)
41	1001	4.350 (1.52)	7.921 (−2.46)	4.425 (3.26)	7.856 (−3.26)
81	1001	4.419 (3.12)	8.031 (−1.10)	4.464 (4.18)	7.994 (−1.55)
161	1001	4.408 (2.87)	8.075 (−0.56)	4.435 (3.50)	8.054 (−0.81)
81	2001	4.422 (3.19)	8.030 (−1.11)	4.464 (4.18)	7.996 (−1.53)
161	4001	4.412 (2.97)	8.074 (−0.57)	4.437 (3.55)	8.055 (−0.80)

Table 5: Comparison of x_1 values at Re =800 with percentage errors in parentheses. The reference value for the reattachment length at the base of the domain is x_1 =6.045, half the reported values. *: Mesh given in testing suite of *hydra-th*; †: Equivalent to Barton's Mesh 7;

N_y	N_x	$x_{1,u_x=0}$	$x_{1,\omega_z=0}$	$x_{1,u_x=0} - x_{1,\omega_z=0}$
101†	201	6.018 (−0.44)	6.008 (−0.61)	1.02×10^{-2}
101	1803	6.012 (−0.55)	6.003 (−0.69)	8.72×10^{-3}
41*	801	5.713 (−5.50)	5.688 (−5.91)	2.50×10^{-2}
41	1001	5.702 (−5.68)	5.677 (−6.09)	2.50×10^{-2}
81	1001	5.977 (−1.13)	5.964 (−1.33)	1.23×10^{-2}
161	1001	6.060 (0.25)	6.054 (0.15)	6.51×10^{-3}
81	2001	5.988 (−0.94)	5.977 (−1.13)	1.13×10^{-2}
161	4001	6.063 (0.29)	6.057 (0.20)	5.95×10^{-3}

Table 6: Comparison of x_2 and x_3 values at Re = 800 with percentage errors in parentheses. The reference values for the separation and reattachment length at the base of the domain are x_2 =4.770 and x_3 =11.105, half the reported values. *: Mesh given in testing suite of *hydra-th*; †: Equivalent to Barton's Mesh 7;

N_y	N_x	$x_{2,u_x=0}$	$x_{3,u_x=0}$	$x_{2,\omega_z=0}$	$x_{3,\omega_z=0}$
101†	201	4.873 (2.16)	10.381 (−6.52)	4.911 (2.96)	10.346 (−6.83)
101	1803	4.868 (2.06)	10.391 (−6.43)	4.900 (2.73)	10.361 (−6.70)
41*	801	4.720 (−1.06)	10.146 (−8.63)	4.789 (0.41)	10.075 (−9.28)
41	1001	4.709 (−1.29)	10.148 (−8.62)	4.779 (0.18)	10.076 (−9.26)
81	1001	4.858 (1.84)	10.361 (−6.70)	4.901 (2.75)	10.320 (−7.07)
161	1001	4.878 (2.27)	10.428 (−6.10)	4.904 (2.81)	10.405 (−6.31)
81	2001	4.869 (2.07)	10.359 (−6.72)	4.909 (2.91)	10.321 (−7.06)
161	4001	4.881 (2.33)	10.428 (−6.10)	4.905 (2.82)	10.407 (−6.29)