ENCCS
EuroCC National Competence Centre Sweden

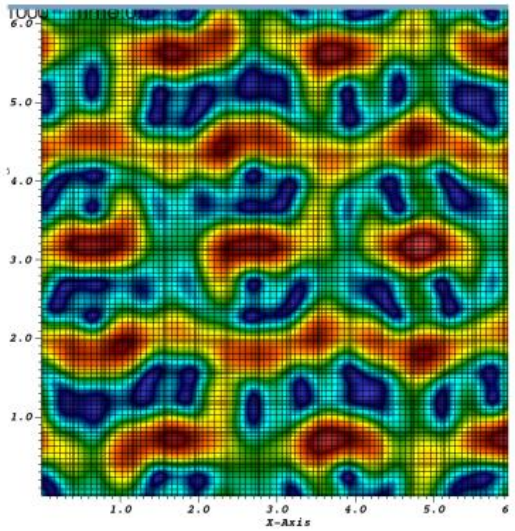
EXCELLERAT

Nek5000 v19;
Hands on session 1

Adam Peplinski

KTH Engineering Mechanics

First run: eddy_uv



Requirements:

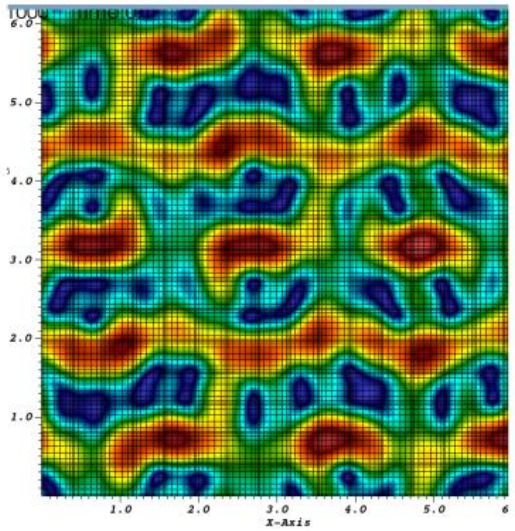
- Nek5000 code
- gmake
- gcc, gfortran
- mpicc, mpifort/mpif77
- VisIt, Paraview
- gmsh

eddy_uv.nek5000

```
filetemplate: eddy_uv%01d.f%05d
firsttimestep: 1
numtimesteps: 11
```

```
tar zxf Nek5000-19.0.tar.gz
cd ./Nek5000/
export NEK_SOURCE_ROOT=`pwd`
PATH=${NEK_SOURCE_ROOT}/bin:$PATH
cd tools/
./maketools all
cd ../run/
cp –r ../examples/eddy_uv/ ./
cd eddy_uv
genmap
            eddy_uv
            0.01
makenek eddy_uv
nekbmpi eddy_uv 4
visnek
```

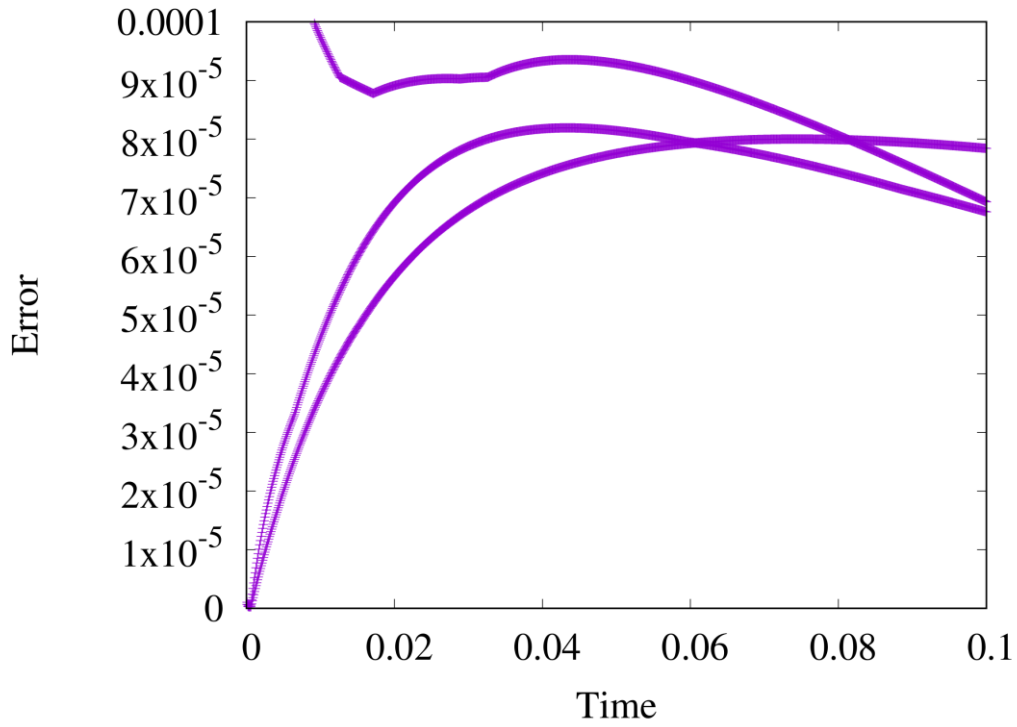
First run: eddy_uv



eddy_uv.nek5000

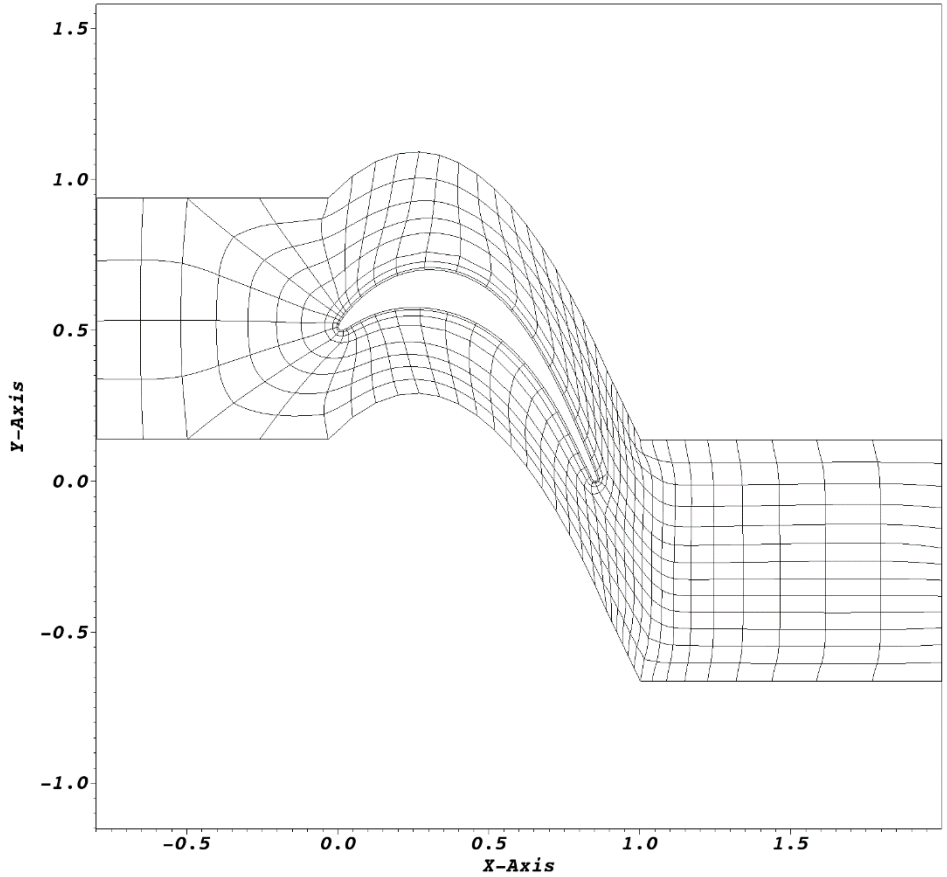
- Eddy solutions in doubly-periodic domain with an additional translational velocity
- Exact solution known
- Error plot

  grep err logfile > err.txt

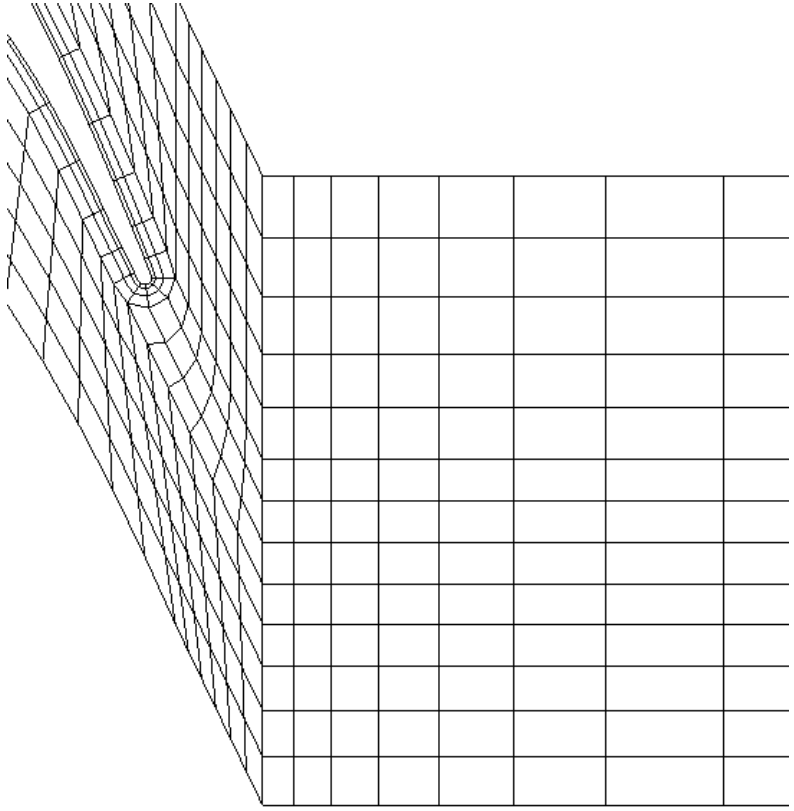


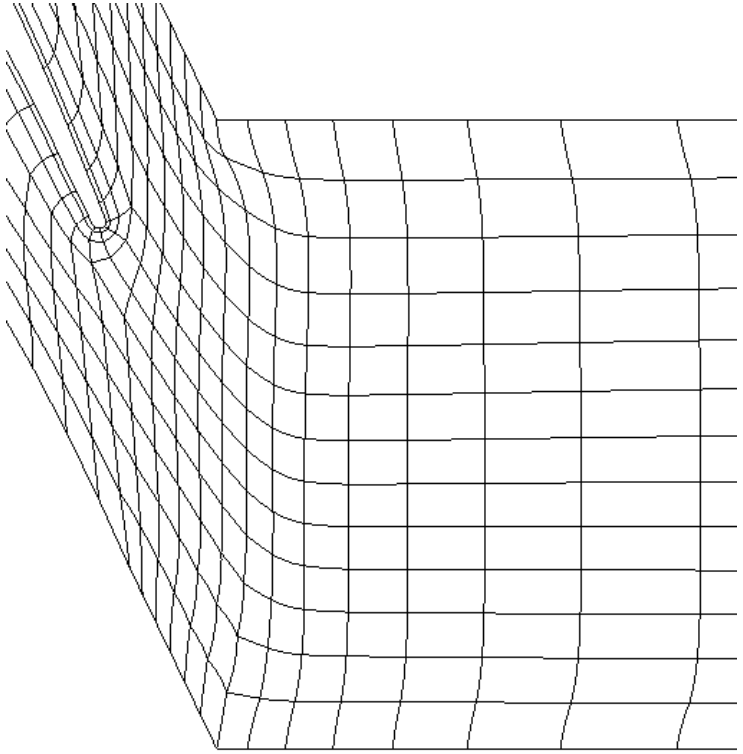
Other interesting examples

- hemi: Particle tracking in flow past a hemispherical roughness.
- lin_channel2D: Linear solver.
- ocyl: Moving meshes.
- smoother: Mesh smoothing
- taylor: Taylor-Couette flow.
- turbChannel: Turbulent channel flow
- vortex: Vortex breakdown in a container with a rotating lid



Mesh smoothing





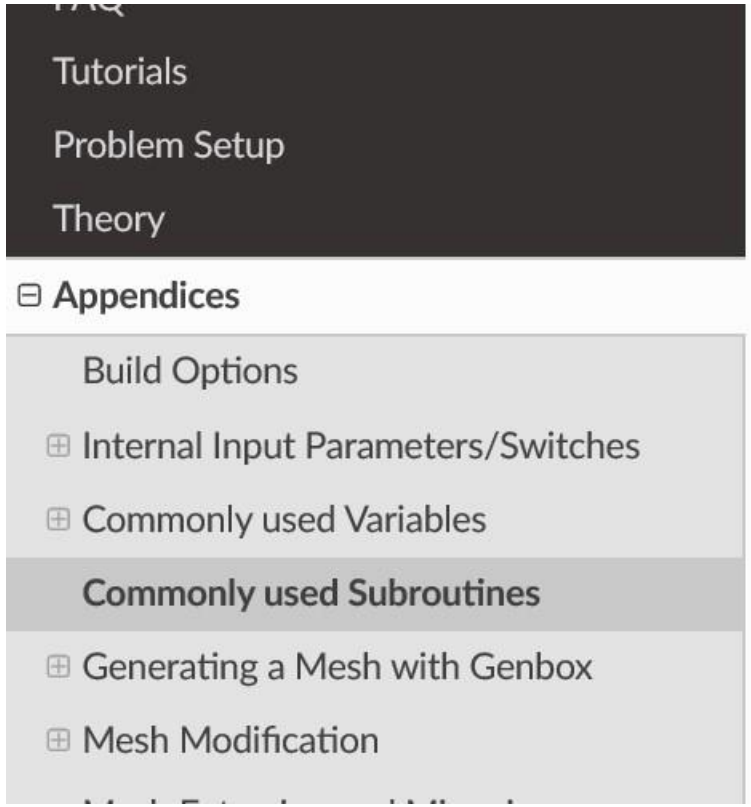
Mesh smoothing


```
#include "experimental/meshsmoother.f"
```

```
c-----------------------------------------------------------
C
C  USER SPECIFIED ROUTINES:
C
C     - boundary conditions
C     - initial conditions
C     - variable properties
C     - local acceleration for fluid (a)
C     - forcing function for passive scalar (q)
C     - general purpose routine for checking errors etc.
C
c-----------------------------------------------------------
      subroutine uservp (ix,iy,iz,eg)

      parameter (init_trans=0)
      include 'SIZE'
```

```
      subroutine usrdat2
      include 'SIZE'
      include 'TOTAL'

      call meshsmoother

ccc   The default parameters for the smoother are:
c      parameter(nbc=1)         !number of boundary conditions
c      character*3 dcbc(nbc)
c      save        dcbc
c      data        dcbc /'W  '/  !BCs listed here

c      idftyp = 0      !distance function - 0 -> exponential, 1-> tanh
c      alpha = 15.     !Input for wall distance function
c      beta  = 0.1     !

c      nouter = 50      !total loops around laplacian and optimizer smoothing
c      nlap = 20        !number of laplacian iterations in each loop
c      nopt = 20        !number of optimization iterations in each loop

c      mtyp = 1         !metric type
ccc
ccc    NOTE:The user can modify these parameters and run the smoother
ccc    with modified parameters as:
ccc    call smoothmesh(mtyp,nouter,nlap,nopt,nbc,dcbc,idftyp,alpha,beta)

      return
      end
```

Common routines



average of `f`. Depending on the value of the logical, `ifverbose`, the results will be printed to standard output along with name. In `avg2`, the `f` component is squared. In `avg3`, vector `g` also contributes to the average calculation.

```
subroutine outpost(x,vy,vz,pr,tz,' ')
```

Dumps the current data of `x`, `vy`, `vz`, `pr`, `tz` to an `.fld` or `.f0????` file for post processing.

```
subroutine platform_timer(ivrb)
```

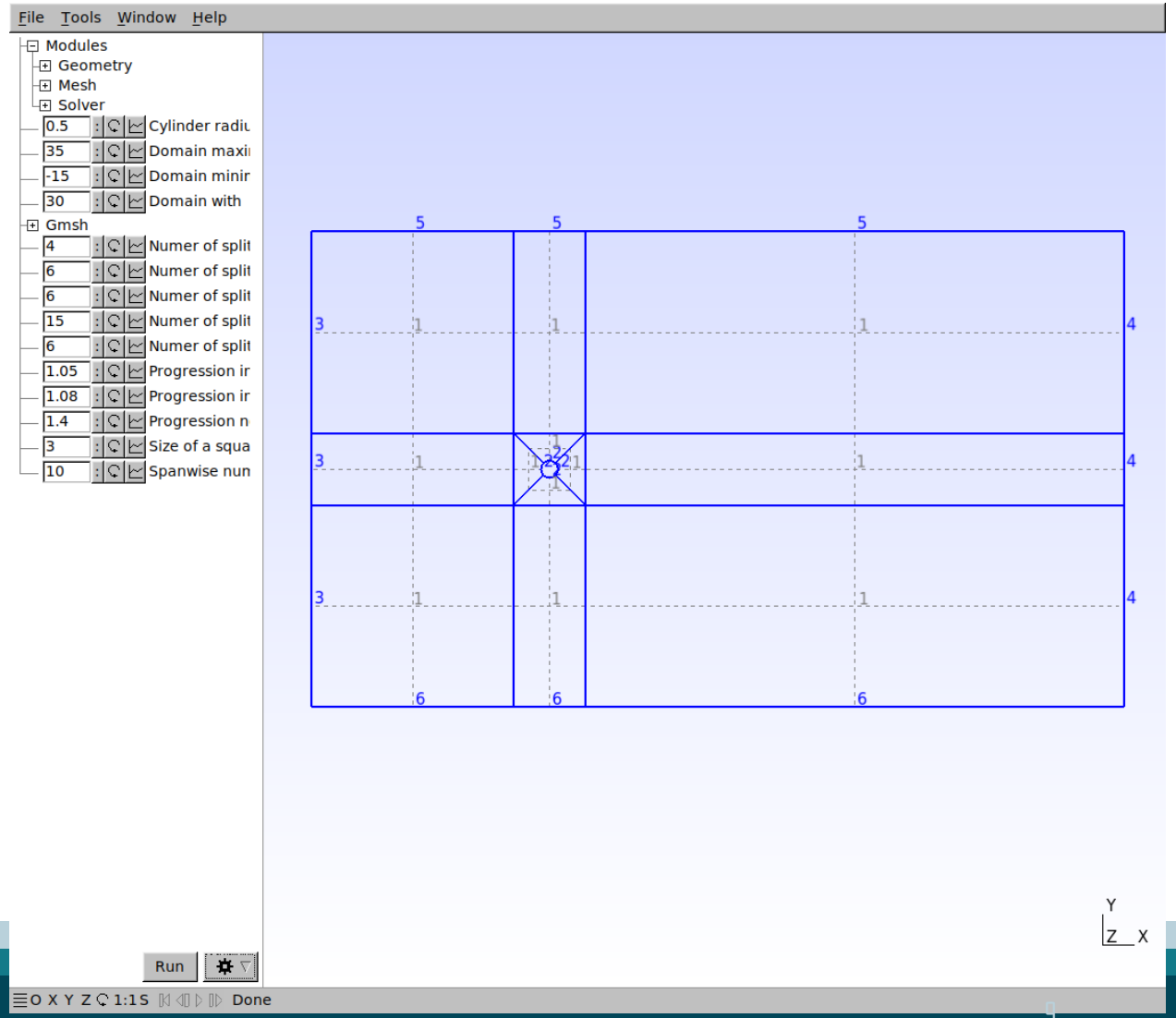
Runs the battery of timing tests for matrix-matrix products,contention-free processor-to-processor ping-pong tests, and `mpi_all_reduce` times. Allows one

Mesh smoothing


```
      subroutine usrdat2
      include 'SIZE'
      include 'TOTAL'

      call outpost(vx,vy,vz,pr,tz,'msh')
      call meshsmoother
      call outpost(vx,vy,vz,pr,tz,'msh')

ccc   The default parameters for the smoother are:
c      parameter(nbc=1)         !number of boundary conditions
c      character*3 dcbc(nbc)
c      save        dcbc
c      data        dcbc /'W  '/  !BCs listed here

c      idftyp = 0      !distance function - 0 -> exponential, 1-> tanh
c      alpha = 15.     !Input for wall distance function
c      beta  = 0.1     !

c      nouter = 50      !total loops around laplacian and optimizer smoothing
c      nlap = 20        !number of laplacian iterations in each loop
c      nopt = 20        !number of optimization iterations in each loop
```

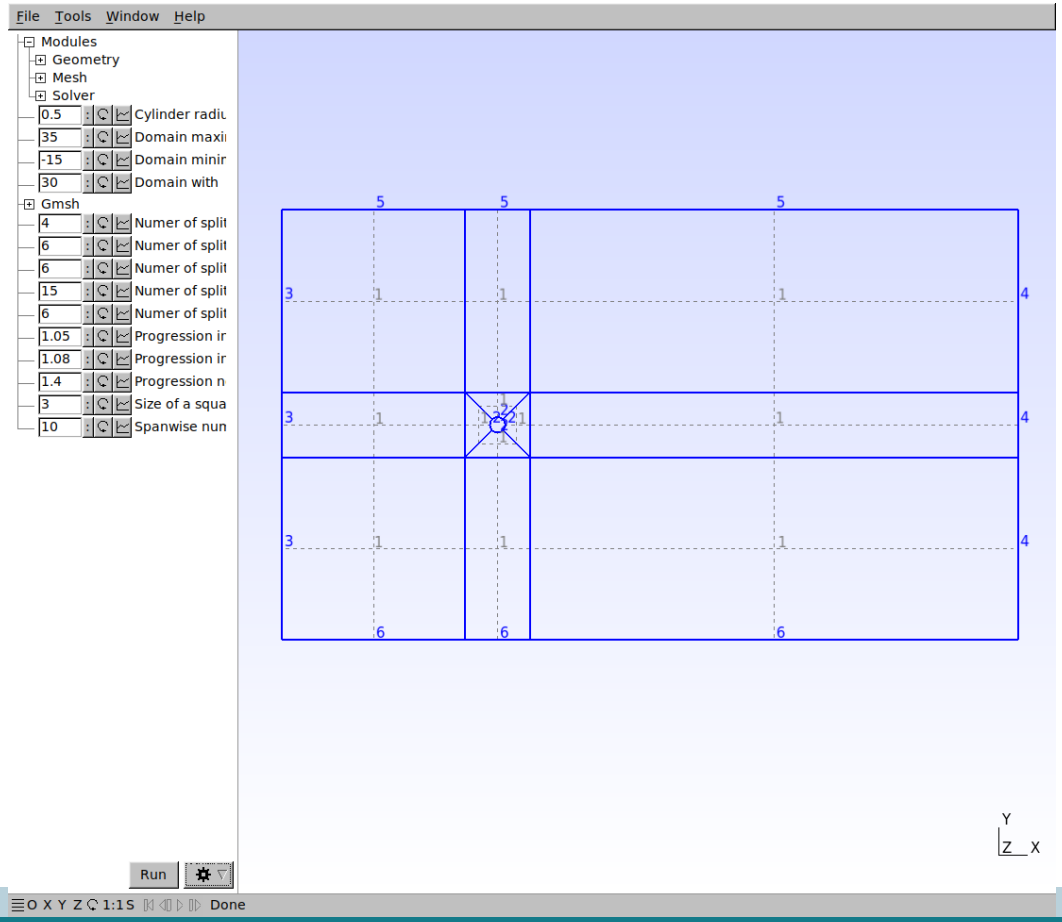
Flow past circular cylinder: **gmsh**

- Advantages:
  - Open source code
  - Existing converter to Nek5000 format: gmsh2nek
  - Flexible scripting language
- Disadvantages:
  - Not a multi-block mesher
  - Lack of three-dimensional recombination algorithm
- Tested with gmsh v 4.8.3



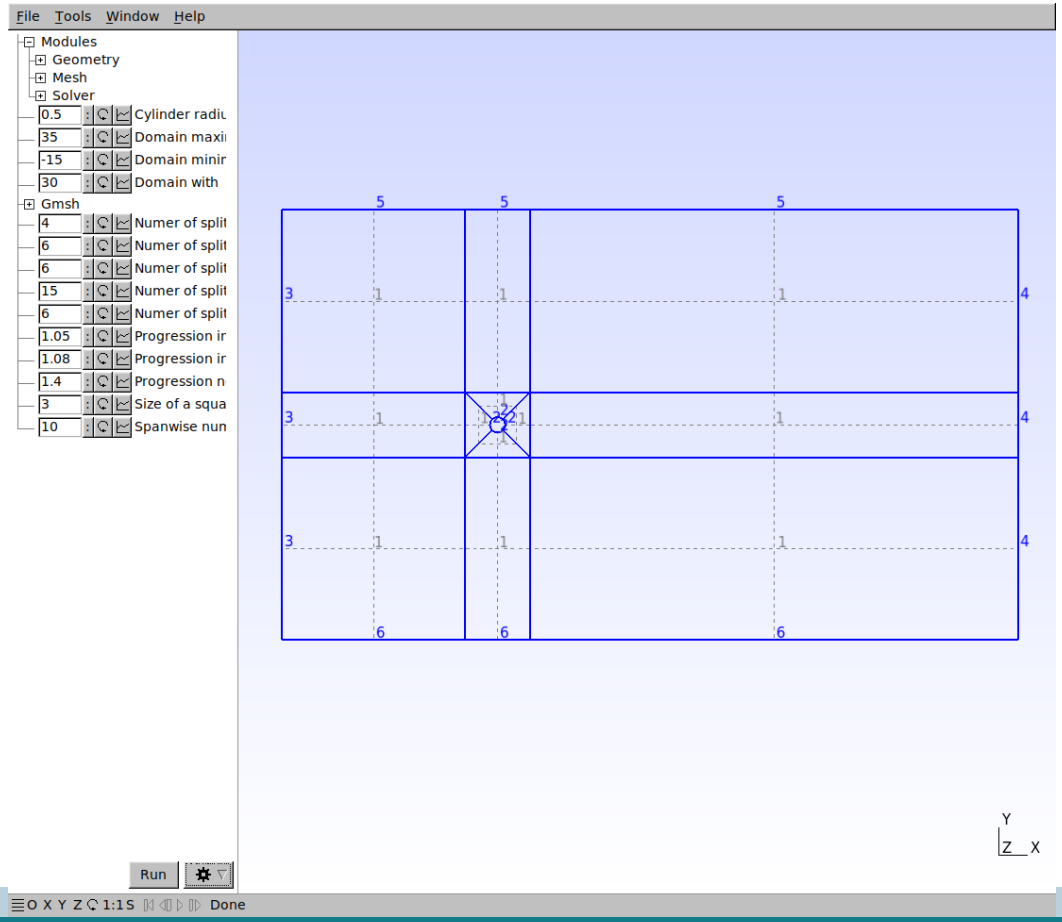
Flow past circular cylinder: **gmsh script**

```
// Parameter section
//////////////////////////////////////////////////////////////////
// I use open cascade
SetFactory("OpenCASCADE");

// mesh parameters
cyl_rad = DefineNumber[ 0.5 , Min 0.25, Max 1.0, Step 0.05, Name "Cylinder radius" ];
box_square = DefineNumber[ 3.0 , Min 2.0, Max 10.0, Step 1.0, Name "Size of a square re
box_min_x = DefineNumber[ -15.0 , Min -30.0, Max -10.0, Step 1.0, Name "Domain minimum
box_max_x = DefineNumber[  35.0 , Min  10.0, Max  50.0, Step 1.0, Name "Domain maximum
box_width = DefineNumber[ 30.0 , Min 20.0, Max 50.0, Step 1.0, Name "Domain with" ];
// mesh parameters
// number of splits at the cylinder surface
srf_nsplit = DefineNumber[ 4 , Min 2, Max 20, Step 1, Name "Numer of splits; cylinder s
// number of splits at the cylinder surface from the wake side
wsrf_nsplit = DefineNumber[ 6 , Min 2, Max 20, Step 1, Name "Numer of splits; cylinder
// number of splits normal to the cylinder surface
nsrf_nsplit = DefineNumber[ 6 , Min 4, Max 20, Step 1, Name "Numer of splits; normal to
// number of splits in front of cylinder
front_nsplit = DefineNumber[ 6 , Min 2, Max 20, Step 1,   Name "Numer of splits; front
// number of splits in wake
wake_nsplit = DefineNumber[ 15 , Min 2, Max 50, Step 1,   Name "Numer of splits; wake"
// spanwise number of splits
spnw_nsplit = DefineNumber[ 10 , Min 2, Max 50, Step 1,   Name "Spanwise numer of split

// progression normal to the cylinder surface
wnprog = DefineNumber[ 1.4 , Min 1, Max 2, Step 0.1,   Name "Progression normal to the
// progression in the wake region
wwprog = DefineNumber[ 1.05 , Min 1, Max 2, Step 0.01,   Name "Progression in the wake
// progression in the front region
wfprog = DefineNumber[ 1.08 , Min 1, Max 2, Step 0.01,   Name "Progression in the front

// Element scale at cylinder surface
cs_el_sc = 0.1;
```



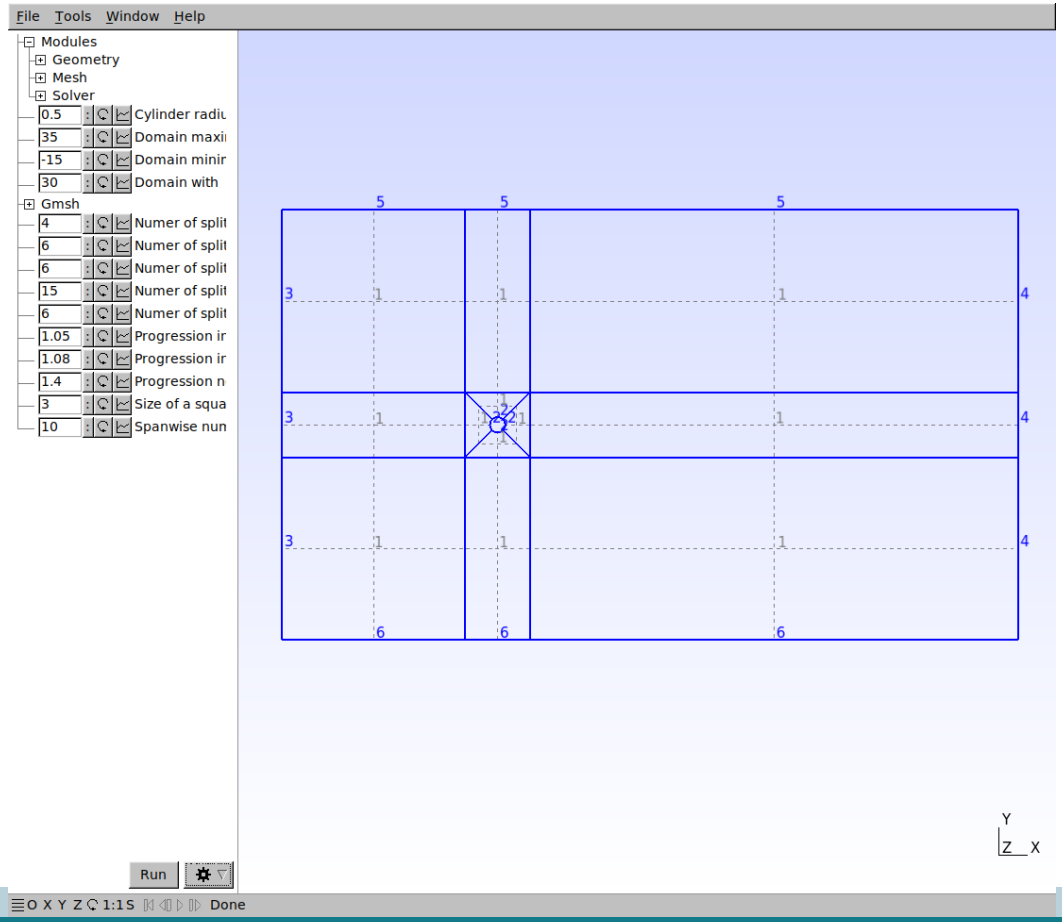
Flow past circular cylinder: **gmsh script**

```
x_pos = cyl_rad*Sin(Pi/4.0);
pts_arc_1 = newp;
Point(newp) = {-x_pos,-x_pos,0.0,cs_el_sc};
pts_arc_2 = newp;
Point(newp) = {-x_pos,x_pos,0.0,cs_el_sc};

x_pos = box_square*Sin(Pi/4.0);
pts_sqr_1 = newp;
Point(newp) = {-x_pos,-x_pos,0.0,cs_el_sc};
pts_sqr_2 = newp;
Point(newp) = {-x_pos,x_pos,0.0,cs_el_sc};

// Curves
list() = {newl};
Circle(newl) = {pts_arc_1,pts_centre,pts_arc_2};
list() += {newl};
Line(newl) = {pts_arc_2,pts_sqr_2};
list() += {newl};
Line(newl) = {pts_sqr_2,pts_sqr_1};
list() += {newl};
Line(newl) = {pts_sqr_1,pts_arc_1};

// Surfaces
surface_list() = {};
crvl = newll;
Curve Loop (crvl) = {list()};
surface_list() += {news};
Surface(news) = {crvl};


// Replicate rotate
langle = 0.5*Pi;
surface_list() += Rotate {{0.0,0.0,1.0},{0.0,0.0,0.0},langle} { Duplicata{ Surface{surface_list(
surface_list() += Rotate {{0.0,0.0,1.0},{0.0,0.0,0.0},-langle} { Duplicata{ Surface{surface_list
surface_list() += Rotate {{0.0,0.0,1.0},{0.0,0.0,0.0},2*langle} { Duplicata{ Surface{surface_list
```



Flow past circular cylinder: **gmsh script**

```
// Remove duplicates
Coherence;

// Extract edges
list() = Unique(Abs(Boundary { Surface{surface_list()}; }));
//For il In {0:# list() - 1}
//   Printf("list %g %g", il,list(il));
//EndFor

////////////////////////////////////////////////////////////////////////
// Physical properties section
////////////////////////////////////////////////////////////////////////
// Volume
Physical Surface("Fluid",1) = {surface_list()};

// Wall
Physical Curve("Wal",2) = {list(3),list(6),list(9),list(11)};

// Inflow
Physical Curve("Inf",3) = {list(14),list(24),list(26)};

// Outflow
Physical Curve("Out",4) = {list(17),list(28),list(30)};

// Top periodic
Physical Curve("Pet",5) = {list(23),list(27),list(31)};

// Bottom periodic
Physical Curve("Peb",6) = {list(20),list(25),list(29)};
```



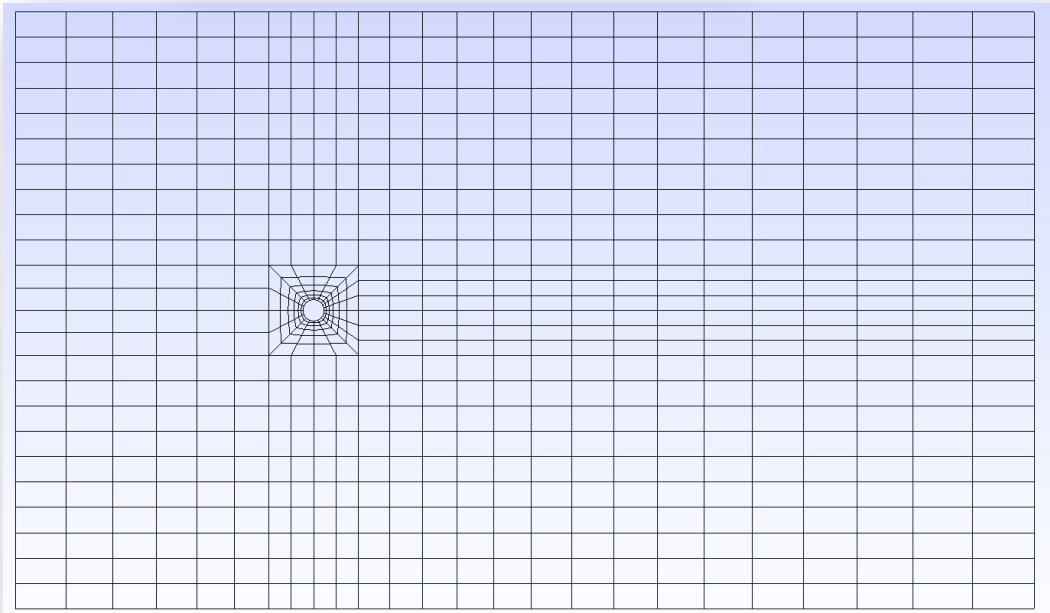
Flow past circular cylinder: **gmsh script**


```
////////////////////////////////////////////////////////////////////
// Transfinite division section
////////////////////////////////////////////////////////////////////
// Edge division
//
Transfinite Curve{list(3),list(6),list(9),list(1),list(5),list(7),list(14),list(20),list(23)} = (srf_nsplit+1) Using Progression 1;

Transfinite Curve{list(11),list(10),list(17)} = (wsrf_nsplit+1) Using Progression 1;

Transfinite Curve{list(18),list(19),list(21),list(22),list(24),list(26),list(28),list(30)} = (spnw_nsplit+1) Using Progression 1;

Transfinite Curve{list(12),list(13),list(25),list(27)} = (front_nsplit+1) Using Progression wfprog;

Transfinite Curve{list(15),list(16),list(29),list(31)} = (wake_nsplit+1) Using Progression wwprog;

Transfinite Curve{-list(0),list(2),-list(4),list(8)} = (nsrf_nsplit+1) Using Progression wnprog;

// Surface division
Transfinite Surface{surface_list()};
Recombine Surface{surface_list()};
```

Flow past circular cylinder: **gmsh script**

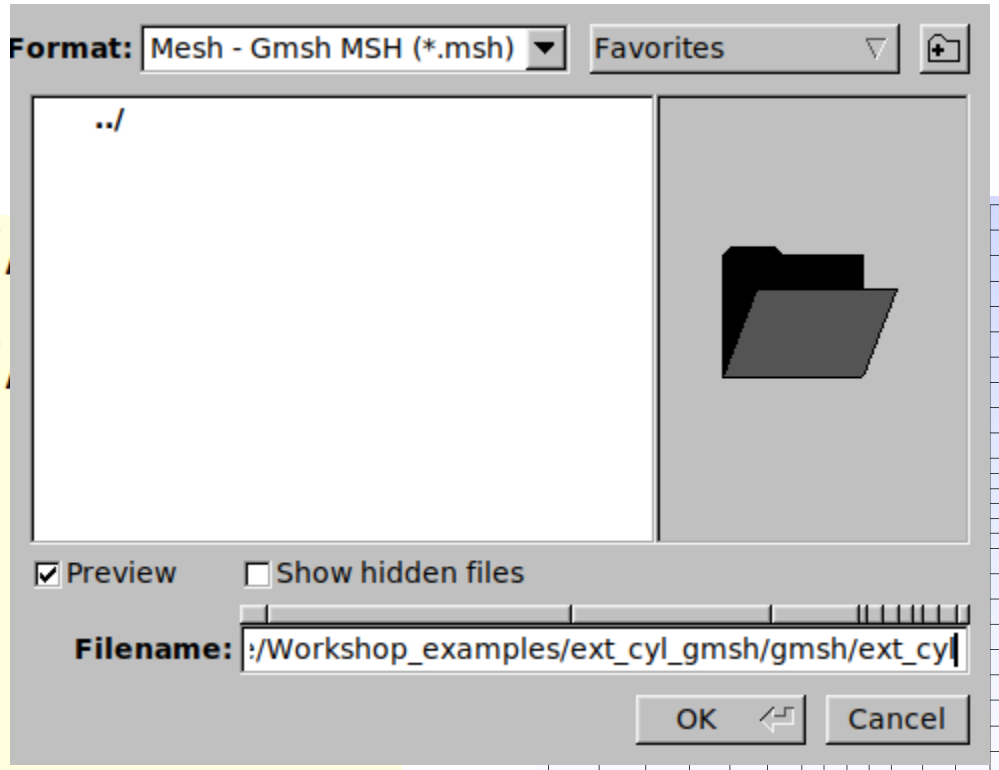
```
//+++++++++++++++++++++++++++++++++++++++++++++
// Meshing section
//+++++++++++++++++++++++++++++++++++++++++++++
Mesh 1;
Mesh 2;
Mesh 3;

SetOrder 2;

RenumberMeshElements;

//+++++++++++++++++++++++++++++++++++++++++++++
// Mesh saving section
//+++++++++++++++++++++++++++++++++++++++++++++
Mesh.Format = 1;
Mesh.MshFileVersion = 2.2;
Mesh.SaveAll = 0;
Mesh.Binary = 0;

Save "ext_cyl.msh";
```

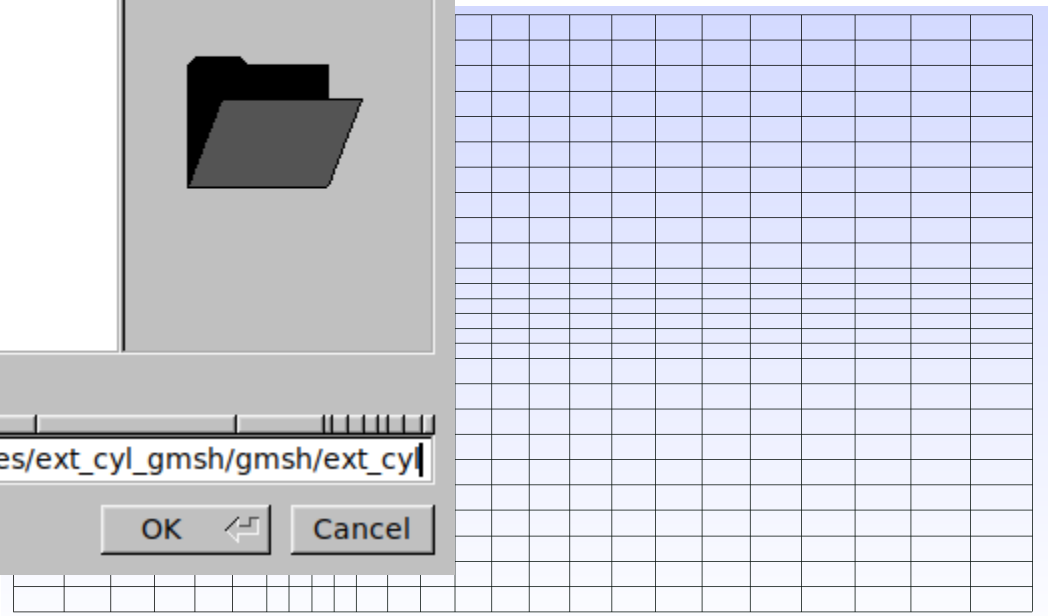
```
#!/bin/bash
# Script to generate meshes
gmsh ext_cyl.geo -
```



Flow past circular cylinder: **gmsh script**

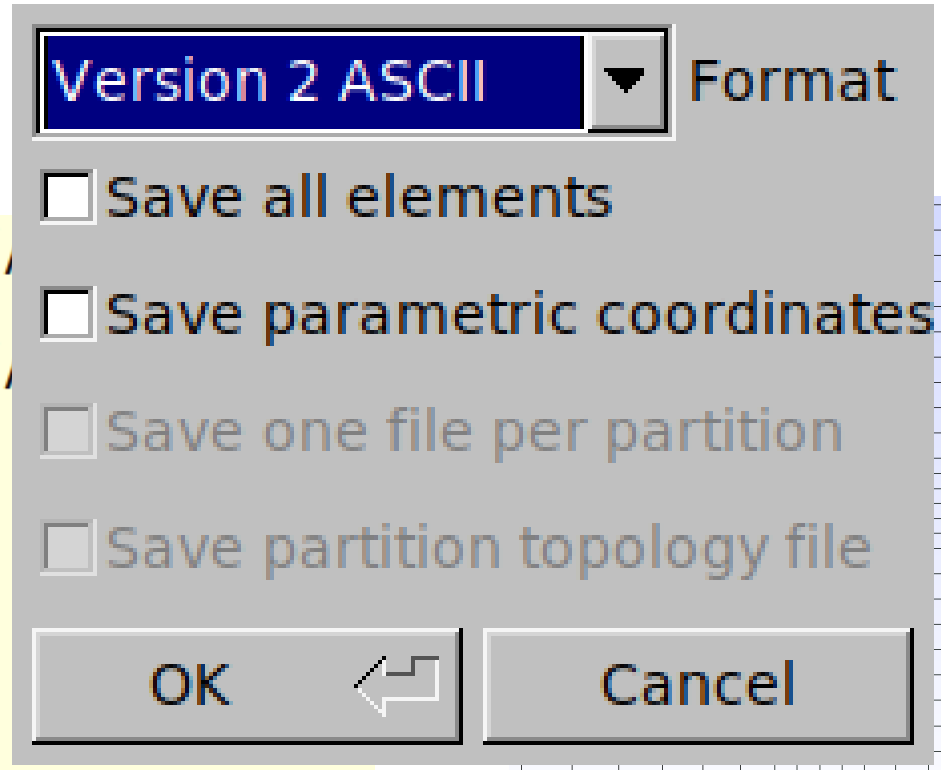
```
//////////////////////////
// Meshing section
//////////////////////////
Mesh 1;
Mesh 2;
Mesh 3;

SetOrder 2;

RenumberMeshElements;
```

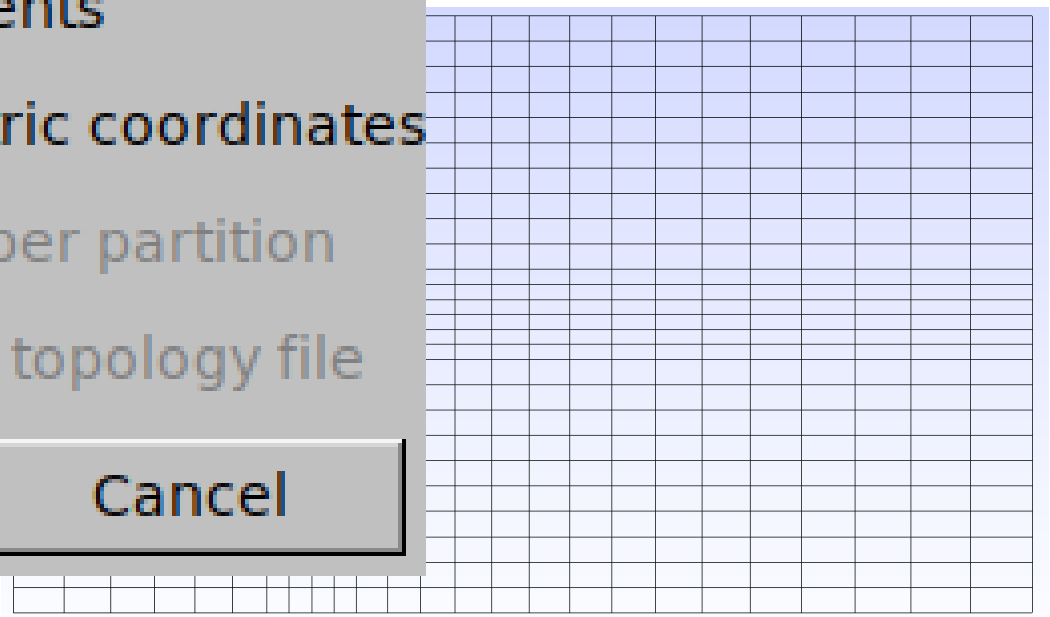




Flow past circular cylinder: **gmsh script**

```
//////////////////////////
// Meshing section
//////////////////////////
Mesh 1;
Mesh 2;
Mesh 3;

SetOrder 2;

RenumberMeshElements;
```





Flow past circular cylinder: **gmsh2nek**

```
skagsnebb[adam]/scratch/adam/ext_cyl_gmsh/gmsh> gmsh2nek
Enter mesh dimension: 2
Input .msh file name: ext_cyl
 total node number is          3006
 total line element number is          118
 total quad element number is          722
 ***************************************************
 Boundary info summary
 BoundaryName     BoundaryID
 Wal           2
 Inf           3
 Out           4
 Pet           5
 Peb           6
 ***************************************************
 Enter number of periodic boundary surface pairs:
1
 input surface 1 and  surface 2  BoundaryID
6 5
 input translation vector (surface 1 -> surface 2)
0 30 0
 ***************************************************
 Please set boundary conditions to all non-periodic boundaries
 in .usr file usrdat2() subroutine
 ***************************************************

writing ext_cyl.re2
```

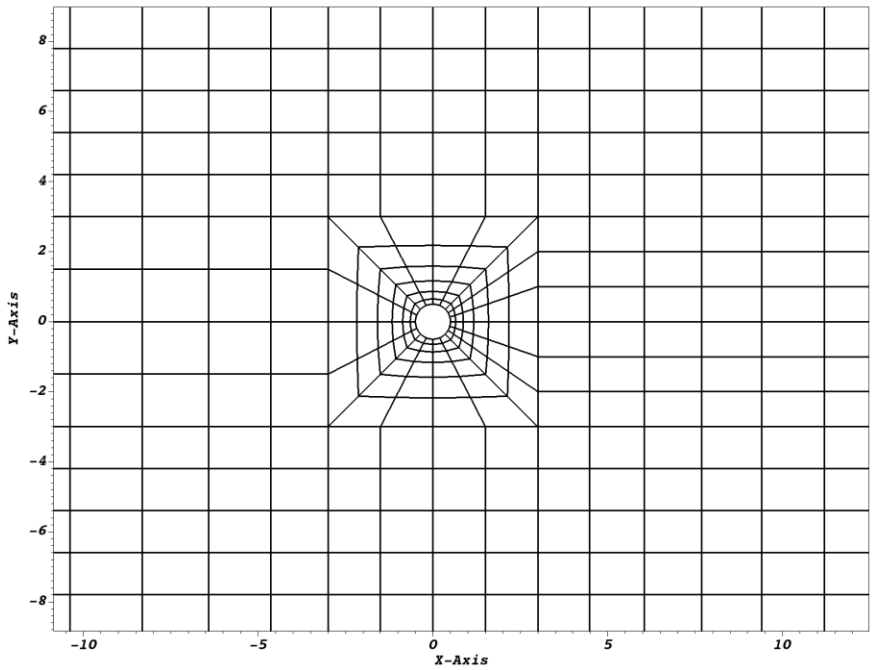
Flow past circular cylinder: **compilation**

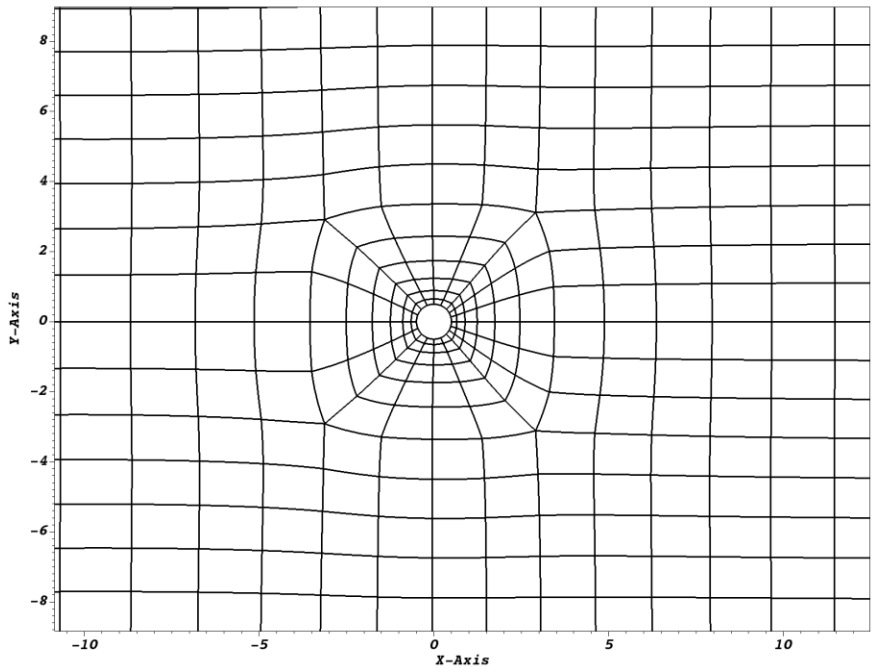

```
./compile_script --clean
./compile_script --all
../../../Nek5000/bin/genmap
../../../Nek5000/bin/nekbmpi ext_cyl 4
../../../Nek5000/bin/visnek
```

```
c     Mesh smoother
#include "experimental/meshsmoother.f"
c-----------------------------------------------------------------------
      subroutine uservp (ix,iy,iz,eg)
      include 'SIZE'
      include 'NEKUSE'

      integer e,f,eg
```

```
c-----------------------------------------------------------------------
      subroutine usrdat

!     Set all non-periodic BCs here. This is required due to generating m
      call setbc(2,1,'W  ') ! wall
      call setbc(3,1,'v  ') ! inflow
      call setbc(4,1,'O  ') ! outflow

      return
      end
c-----------------------------------------------------------------------
      subroutine usrdat2
      include 'SIZE'
      include 'SOLN'

      call outpost(vx,vy,vz,pr,t,'smh')
      call meshsmoother
      call outpost(vx,vy,vz,pr,t,'smh')

      return
      end
c-----------------------------------------------------------------------
      subroutine usrdat3

      return
      end
c-----------------------------------------------------------------------
```

Flow past circular cylinder: **mesh smoothing**





Flow past circular cylinder: **userchk**


```
      subroutine userchk
      include 'SIZE'
      include 'INPUT'
      include 'TSTEP'

!     object definition
      integer iobj_wall, bIDs(1)
      save iobj_wall

!     Relative point for torque calculation
      real x0(3)
      save x0
      data x0 /3*0/

      if (istep.eq.0) then      ! initialisation step
         bIDs(1) = 2            ! 'W  '
         call create_obj(iobj_wall,bIDs,1) ! define objects for surface
      else

         call estimate_strouhal

         scale = 2.             ! Cd = F/(.5 rho U^2 ) = 2*F
!     get torques
         ioutpt = int(UPARAM(1))
         if (mod(istep,ioutpt).eq.0)
     $        call torque_calc(scale,x0,.true.,.false.)

!     write out probes
         if (mod(ISTEP,ioutpt).eq.0) call hpts()
      endif

      return
      end
```

```
c-----------------------------------------------------------------------
      subroutine usrdat

!     Set all non-periodic BCs here. This is required due to generating m
      call setbc(2,1,'W  ') ! wall
      call setbc(3,1,'v  ') ! inflow
      call setbc(4,1,'O  ') ! outflow

      return
      end
c-----------------------------------------------------------------------
      subroutine usrdat2
      include 'SIZE'
      include 'SOLN'

      call outpost(vx,vy,vz,pr,t,'smh')
      call meshsmoother
      call outpost(vx,vy,vz,pr,t,'smh')

      return
      end
c-----------------------------------------------------------------------
      subroutine usrdat3

      return
      end
c-----------------------------------------------------------------------
```

Flow past circular cylinder: **userchk**

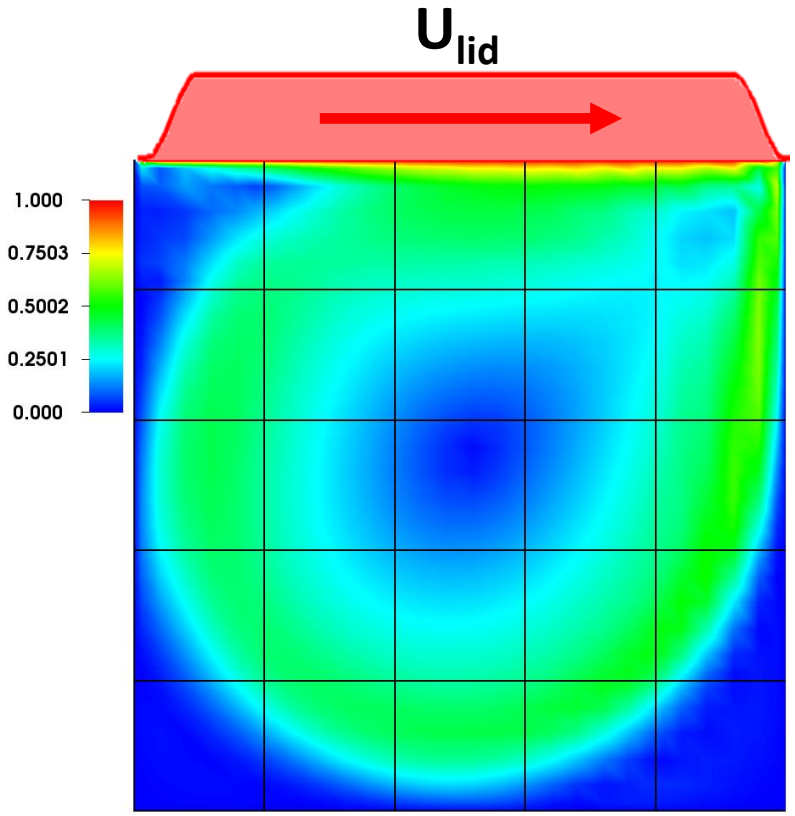

```
      subroutine userchk
      include 'SIZE'
      include 'INPUT'
      include 'TSTEP'

!     object definition
      integer iobj_wall, bIDs(1)
      save iobj_wall

!     Relative point for torque calculation
      real x0(3)
      save x0
      data x0 /3*0/

      if (istep.eq.0) then      ! initialisation step
         bIDs(1) = 2            ! 'W  '
         call create_obj(iobj_wall,bIDs,1) ! define objects for surface
      else

         call estimate_strouhal

         scale = 2.             ! Cd = F/(.5 rho U^2 ) = 2*F
!     get torques
         ioutpt = int(UPARAM(1))
         if (mod(istep,ioutpt).eq.0)
     $        call torque_calc(scale,x0,.true.,.false.)

!     write out probes
         if (mod(ISTEP,ioutpt).eq.0) call hpts()
      endif

      return
      end
```

```
[GENERAL]
#startFrom = restart.fld
stopAt = numSteps #endTime
#endTime = 0.00000
numSteps = 102

userparam01 = 10

dt = 1.000000E-02
timeStepper = bdf3
variableDt = no
targetCFL = 1.00000

writeControl = timeStep #runTime
writeInterval = 20.0

dealiasing = yes
filtering = explicit
filterWeight = 0.100000E-01
filterCutoffRatio = 0.84
```

Flow past circular cylinder


```
c-----------------------------------------------------------------------
      subroutine estimate_strouhal

      include 'SIZE'
      include 'TOTAL'

      real tlast,vlast,tcurr,vcurr,t0,t1
      save tlast,vlast,tcurr,vcurr,t0,t1
      data tlast,vlast,tcurr,vcurr,t0,t1 / 6*0 /

      integer e,eg,eg0,e0

      eg0 = 622          ! Identify element/processor in wake; my
      mid = gllnid(eg0)
      e0  = gllel (eg0)

      st  = 0

      if (nid.eq.mid) then

         tlast = tcurr
         vlast = vcurr

         tcurr = time
         vcurr = vy (1,ny1,1,e0)

         xcurr = xm1(1,ny1,1,e0)
         ycurr = ym1(1,ny1,1,e0)

         write(6,2) istep,time,vcurr,xcurr,ycurr
    2    format(i9,1p4e13.5,' vcurr')

         if (vlast.gt.0.and.vcurr.le.0) then ! zero crossing w/ ne
            t0  = t1
            t1  = tlast + (tcurr-tlast)*(vlast-0)/(vlast-vcurr)
            per = t1-t0
            if (per.gt.0) st = 1./per
         endif
       endif

       st = glmax(st,1)

       n  = nx1*ny1*nz1*nelv
       ux = glamax(vx,n)
       uy = glamax(vy,n)

       if (nid.eq.0.and.st.gt.0) write(6,1) istep,time,st,ux,uy
     1 format(i5,1p4e12.4,' Strouhal')

       return
       end
c-----------------------------------------------------------------------
```

Exercise : lid-driven cavity

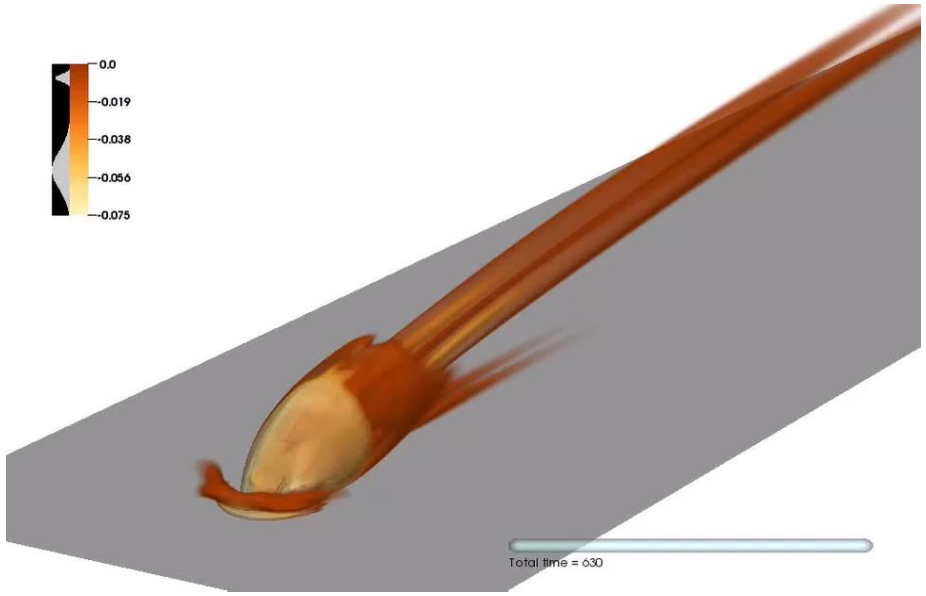



```
!! @param[in] x       function argument
!! @return math_stepf
      real function math_stepf(x)
      implicit none

      ! argument list
      real x

      ! local variables
      real xdmin, xdmax
      parameter (xdmin = 0.001, xdmax = 0.999)
!-----------------------------------------------------------------------
      ! get function vale
      if (x.le.xdmin) then
         math_stepf = 0.0
      else if (x.le.xdmax) then
         math_stepf = 1./( 1. + exp(1./(x - 1.) + 1./x) )
      else
         math_stepf = 1.
      end if

      return
      end function math_stepf
```

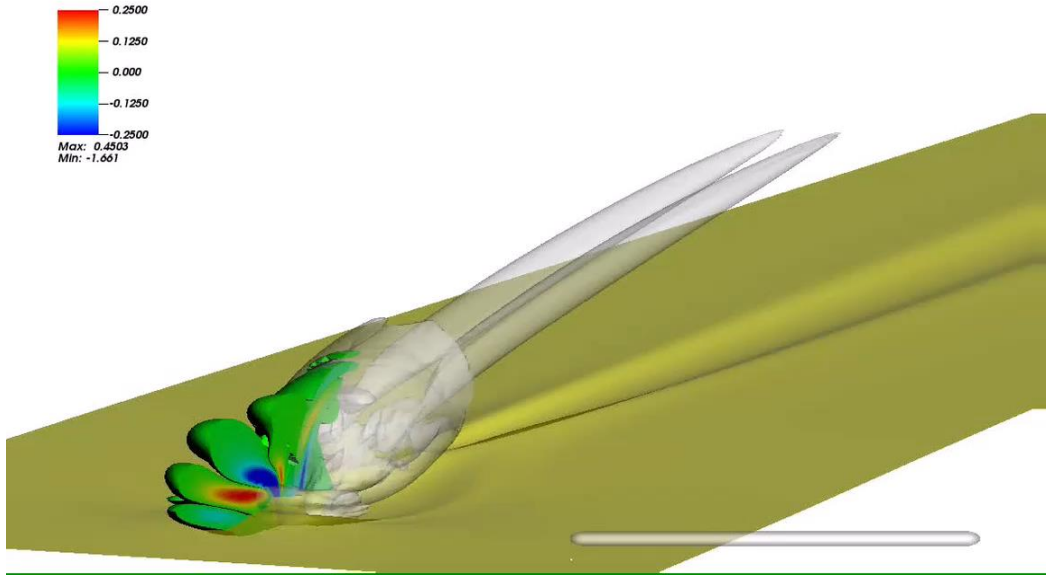
Linear problem

DNS for jet in cross-flow:

Non-linear



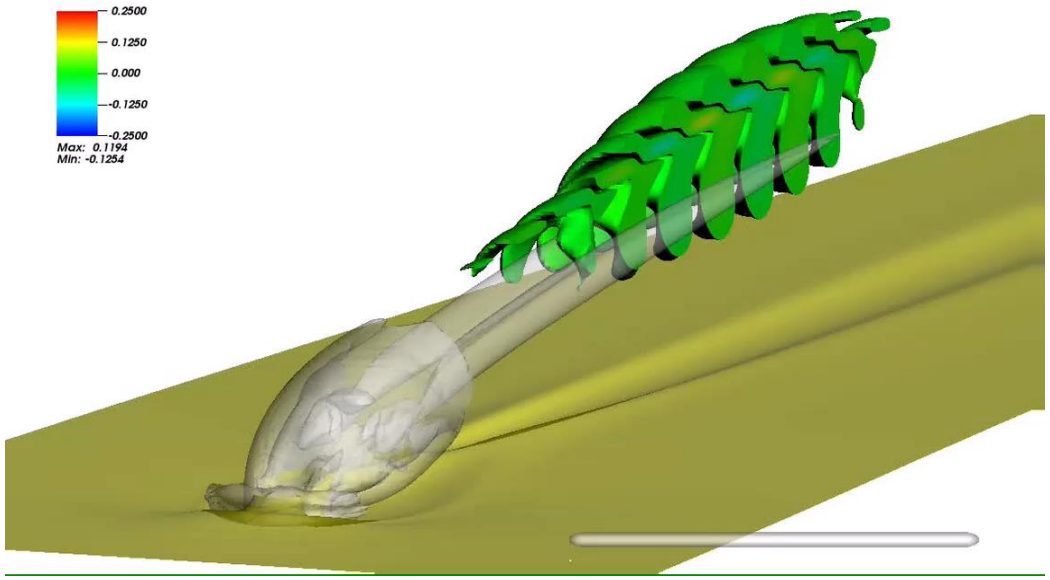
Linear



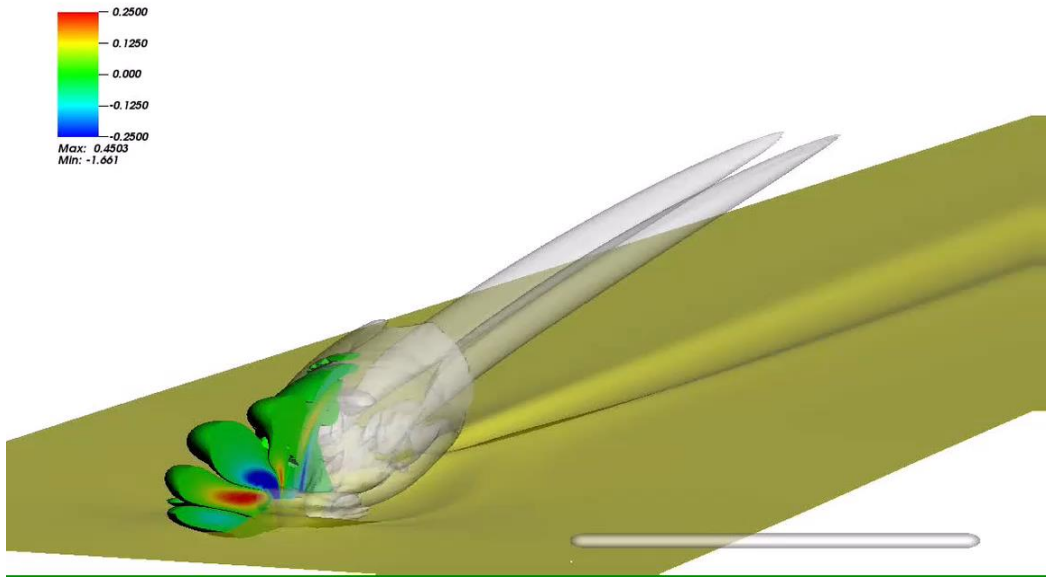
Linear problem

Linear DNS for jet in cross-flow:

Adjoint



Direct



Linear problem: lin_channel2D

```
-rw-r--r-- 1 adam adam   432 Dec  2 12:33 lin_chan_adj.par
-rw-r--r-- 1 adam adam  5348 Dec  2 12:33 lin_chan_adj.re2
-rw-r--r-- 1 adam adam   428 Dec  2 12:33 lin_chan_dir.par
-rw-r--r-- 1 adam adam  5348 Dec  2 12:33 lin_chan_dir.re2
-rw-r--r-- 1 adam adam  5264 Dec  2 12:33 lin_chan.usr
-rw-r--r-- 1 adam adam 96328 Dec  2 12:33 prtlin_chan_adj0.restart
-rw-r--r-- 1 adam adam 96328 Dec  2 12:33 prtlin_chan_dir0.restart
-rw-r--r-- 1 adam adam   451 Dec  2 12:33 README.md
-rw-r--r-- 1 adam adam  2088 Dec  2 12:33 SIZE
```

```
cp -r ../examples/lin_channel2D/ ./
cd lin_channel2D/
../../bin/makenek lin_chan
../../bin/genmap
     lin_chan_dir
     0.01
mkdir run_local
cp lin_chan_dir.* run_local/
cp prtlin_chan_dir0.restart run_local/
cp nek5000 run_local/
cd run_local/
../../../bin/nekbmpi lin_chan_dir 4
../../../bin/visnek
grep 'Energy' logfile > energy.txt
```

lin_channel2D: **SIZE**


```
! BASIC
parameter (ldim=2)               ! domain dimension (2 or 3)
parameter (lx1=10)                ! p-order (avoid uneven and val
parameter (lxd=15)               ! p-order for over-integration (
parameter (lx2=lx1-2)            ! p-order for pressure (lx1 or l

parameter (lelg=50)           ! max total number of elements
parameter (lpmin=1)              ! min MPI ranks
parameter (lpmax=1024)           ! max MPI ranks
parameter (ldimt=1)              ! max auxiliary fields (temperat

! OPTIONAL
parameter (ldimt_proj=1)         ! max auxiliary fields residual
parameter (lhis=1000)            ! max history points
parameter (maxobj=4)             ! max number of objects
parameter (lpert=3)              ! max perturbation modes
parameter (toteq=1)              ! max number of conserved scalar
```

Only $\boldsymbol{P_N - P_{N-2}}$

```
parameter (maxobj=4)              ! max number of objects
parameter (lpert=3)               ! max perturbation modes
parameter (toteq=1)               ! max number of conservec
parameter (nsessmax=1)            ! max sessions
parameter (lxo=lx1)               ! max grid size on output
parameter (mxprev=20,lgmres=30)   ! max dim of projection &
parameter (lorder=3)              ! max order in time
parameter (lx1m=1)              ! polynomial order mesh sc
parameter (lfdm=0)                ! set to 1 for fast diagc
parameter (lelx=1,lely=1,lelz=1) ! global tensor mesh dime

parameter (lelt=lelg/lpmin + 4)  ! max number of local ele
parameter (lbelt=1)               ! mhd
parameter (lpelt=lelt)            ! linear stability
parameter (lcvelt=1)              ! cvode

! INTERNALS
include 'SIZE.inc'
```

lin_channel2D: user interface

```
c      get energy
       domega = 1.0
       nit = 10
       if (mod(ISTEP,nit).eq.0) then
          n = NX1*NY1*NZ1*NELV
          Ek(2) = Ek(1)
          timel(2) = timel(1)
          omega(2) = omega(1)
          Ek(1) = 0.5*(glsc3(VXP,VXP,BM1,n)+glsc3(VYP,VYP,BM1,n))
          timel(1) = TIME
c      get growthrate
          if (Ek(2).gt.0.0.and.timel(1).gt.timel(2)) then
             omega(1) = 0.5*log(Ek(1)/Ek(2))/(timel(1)-timel(2))
             domega = abs((omega(1)-omega(2))/omega(1))
          endif
c      set logs
          if (NIO.eq.0.and.ISTEP.gt.2*nit)
     $       write(*,*) 'Energy ',Ek(1),omega(1),domega,TIME
       endif

c      converged field
       if (ISTEP.eq.NSTEPS) then
c      write perturbation field
          call outpost2(VXP,VYP,VZP,PRP,TP,0,'prt')
       endif
```

```
      subroutine useric (ix,iy,iz,ieg)

      include 'SIZE'
      include 'NEKUSE'          ! UX, UY, UZ, TEMP, Z
      include 'SOLN'            ! JP

      real amp, ran

c     velocity
c     base flow
      if (JP.eq.0) then
         UX = (1.0-Y**2)
         UY = 0.0
         UZ = 0.0
      else
c     perturbation
c     random distribution
         amp = 1.0
         ran = 3.e4*(ieg+X*sin(Y)) - 1.5e3*ix*iy + .5e5*ix
         ran = 1.e3*sin(ran)
         ran = 1.e3*sin(ran)
         ran = cos(ran)
         UX  = ran*amp

         ran = 2.3e4*(ieg+X*sin(Y)) + 2.3e3*ix*iy - 2.e5*ix
         ran = 1.e3*sin(ran)
         ran = 1.e3*sin(ran)
         ran = cos(ran)
         UY  = ran*amp

         UZ = 0.0
      endif

      return
      end
```

lin_channel2D: runtime parameters

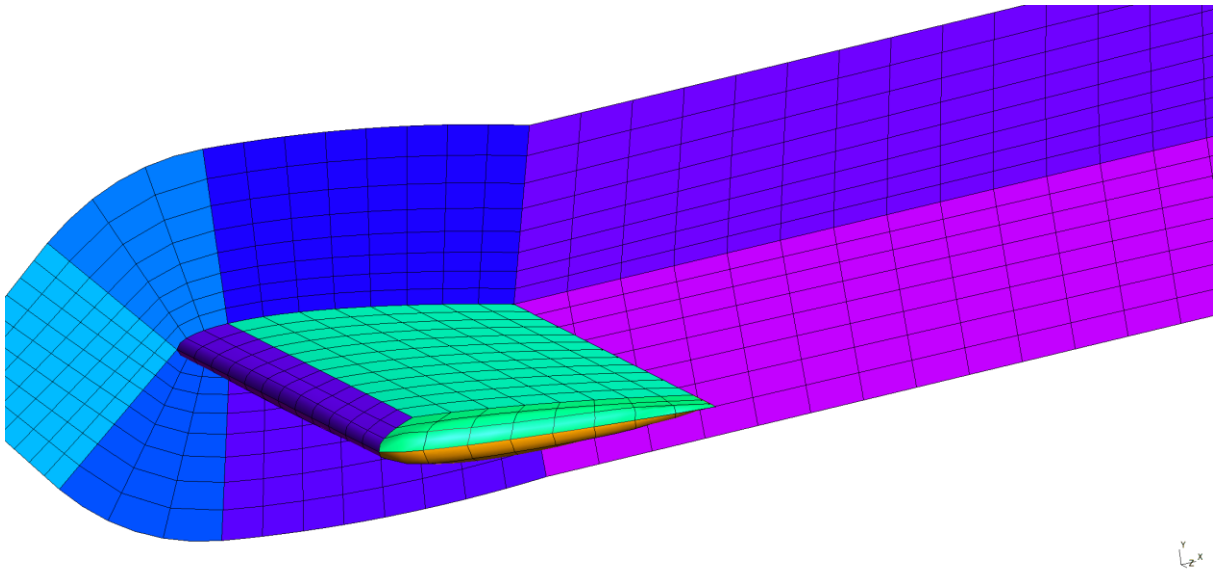
```
[GENERAL]
stopat = numSteps #endTime
numsteps = 1000

userparam01 = 1

dt = -1.0e-02
timestepper = bdf2
writecontrol = timeStep #runTime
writeinterval = 90000
dealiasing = yes

[PROBLEMTYPE]
equation = incompLinNS # incompLinNS
numberOfPerturbations = 1
solveBaseflow = no
variableProperties = no

[PRESSURE]
residualtol = 1e-08
residualproj = no

[VELOCITY]
residualtol = 1e-08
residualproj = no
density = 1
viscosity = -2000.0
```

Meshing with gmsh

NEK5000 – C1U1: NACA0012 aerofoil with rounded tip

- Nek5000-R3: **Efficient strategies for hex-based meshing of complex geometries**
- The goal was to gain user experience finding possible software limitations and workarounds:
  - High-order meshes suitable for **Spectral Element Method**
  - Starting with open source software **gmsh**
  - Focus on proper **representation of object surfaces** and division of the computational domain into a set of **hexahedral subdomains**
  - **Coarse, conforming** mesh as a starting point for AMR

Meshing with gmsh

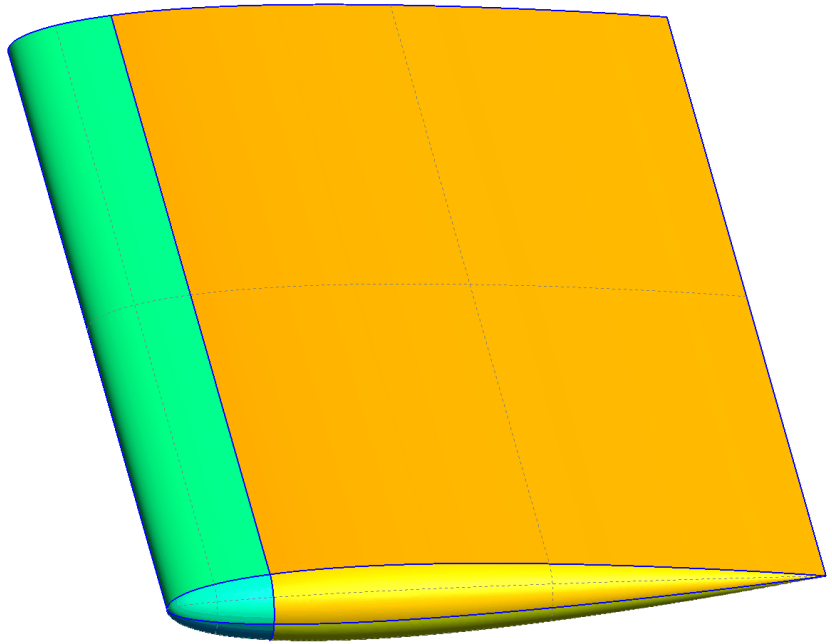


Division in hexahedral subdomains:

- Manual split into volumes
- Use of *Transfinite* algorithm: defined split of volume edges gives surface and volume mesh
- Lack of clear way of edge properties inheritance (duplication, coherence)
- Bottom-up approach (all edger first)
- Labour-intensive; work with low level objects

Meshing with gmsh

Built-in kernel; use of:
Extrude

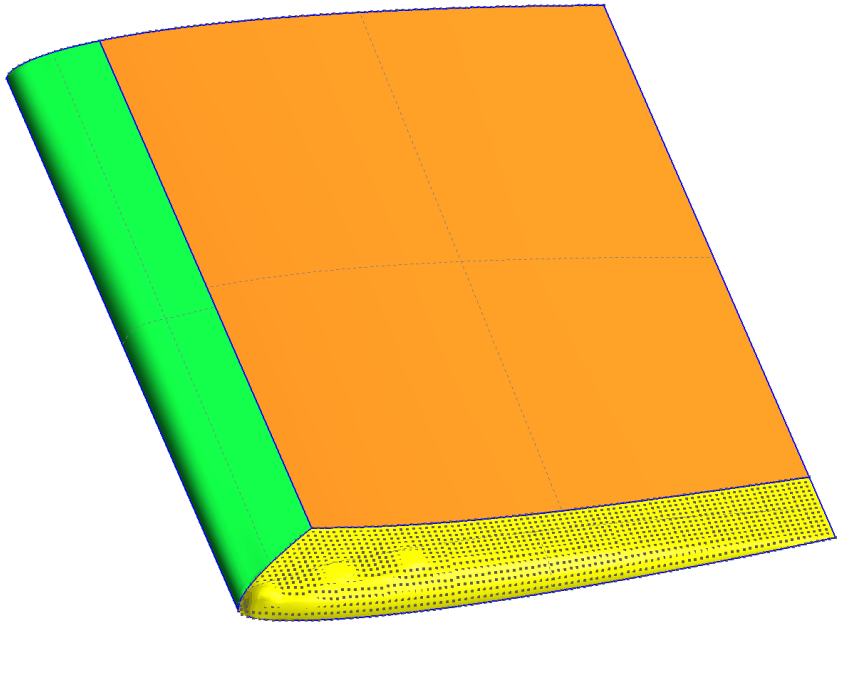


Representation of Object surfaces:

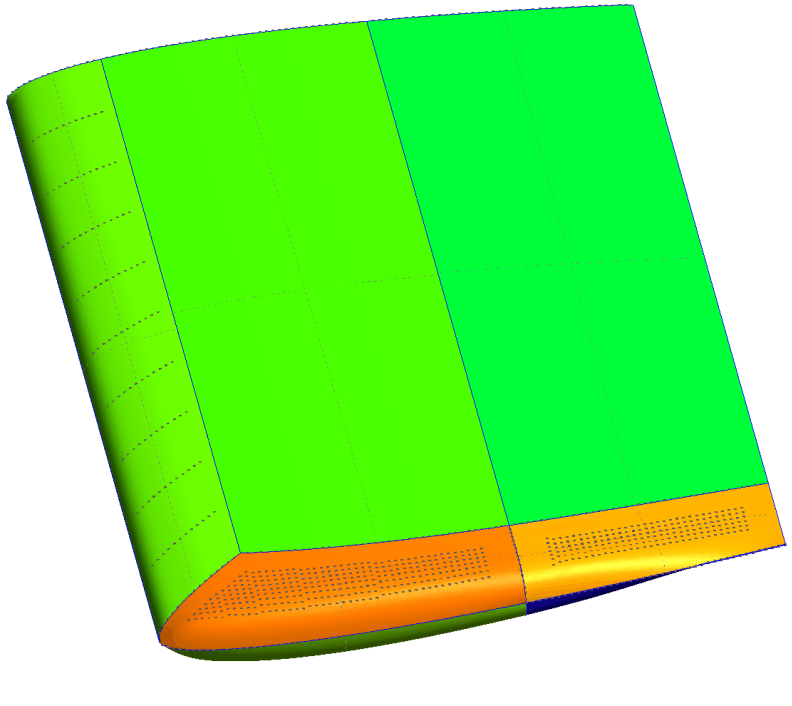
- Built-in kernel not very efficient
  - *Curve{list()} In Surface{};*
  - Not flexible split of sub-surfaces
  - Not useful for generating coarse meshes
- OpenCASCADE kernel more flexible
  - Possibility to immerse points or curves
  - Visible surface corrugations

Meshing with gmsh

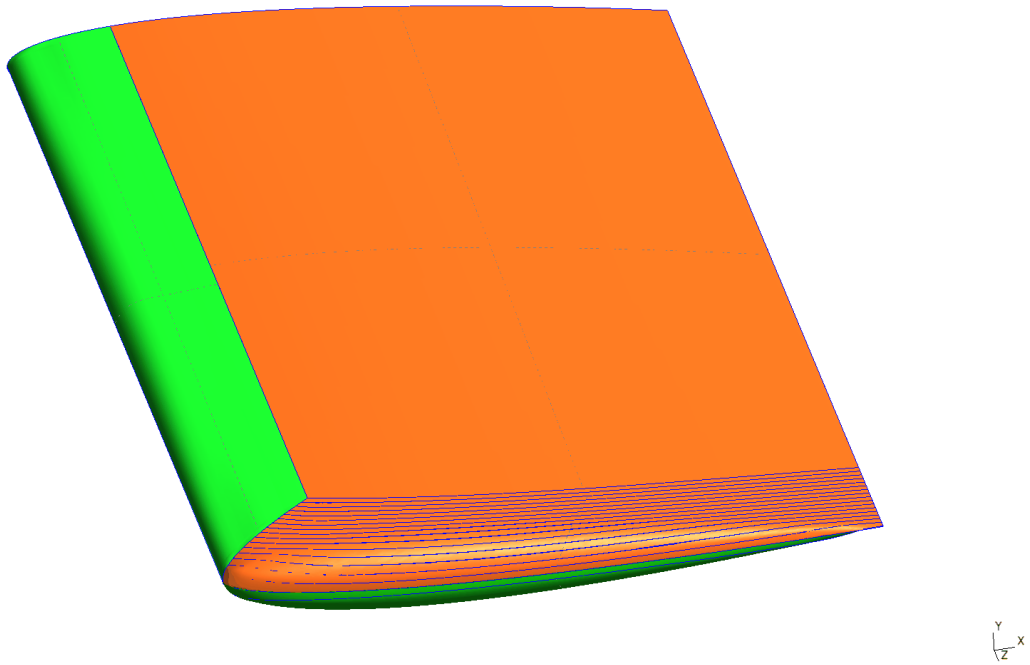
OpenCASCADE kernel; use of:
Surface(tip_srf_up) = {crvl} Using Point{pts_list()};



Adjusted sub-surfaces size, shape and immersed points distribution



Meshing with gmsh



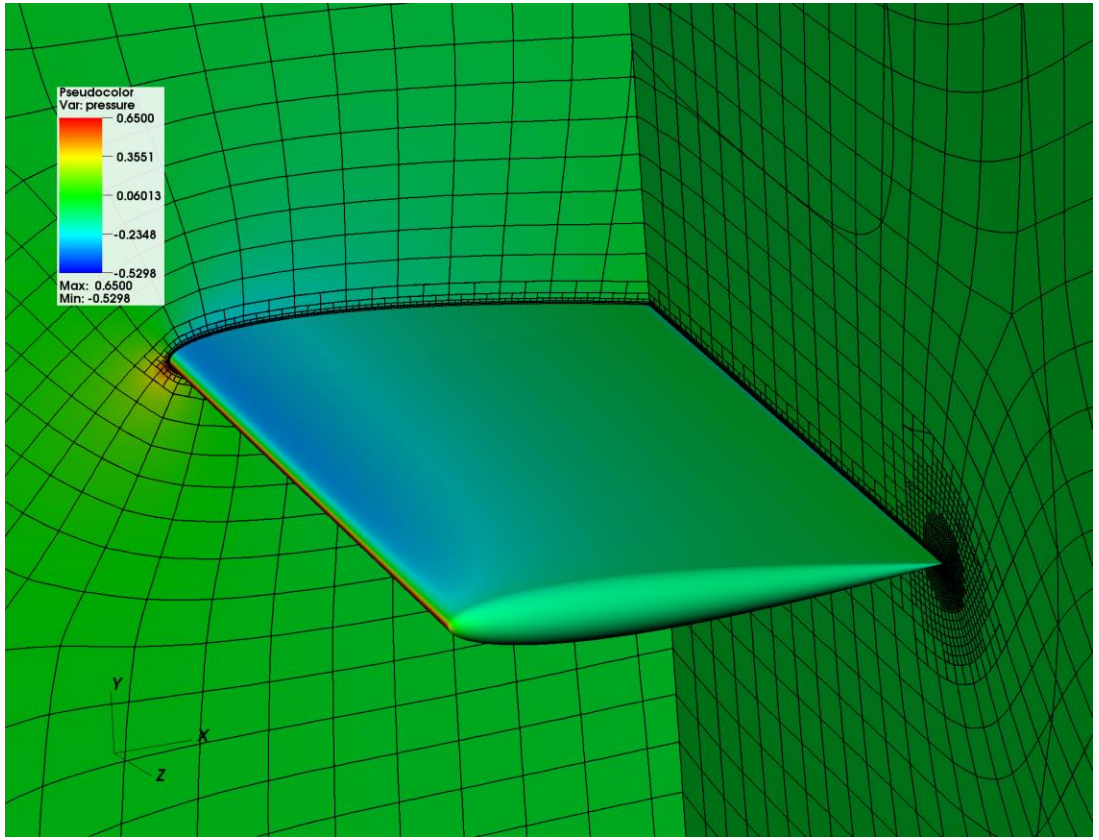
OpenCASCADE kernel:

Wire(newll) = crv_list(0);

srf_tip() = ThruSections{crvi:crve};

- Most exact representation of the surface
- New generated surface edges make *Coherence* operation expensive
- Different mesh parts generated by different scripts

Meshing with gmsh



Mesh generation workflow:

- Generation of inner and bounding box meshes with gmsh
- Exporting to required format
- Conversion to Nek5000 format
- Smoothing inner mehs with Nek5000 smoother
- Merging inner and outer meshes with Nek5000 tools
- Smoothing the resulting mesh