

An informal introduction to NEKNEK

model set-up tutorial

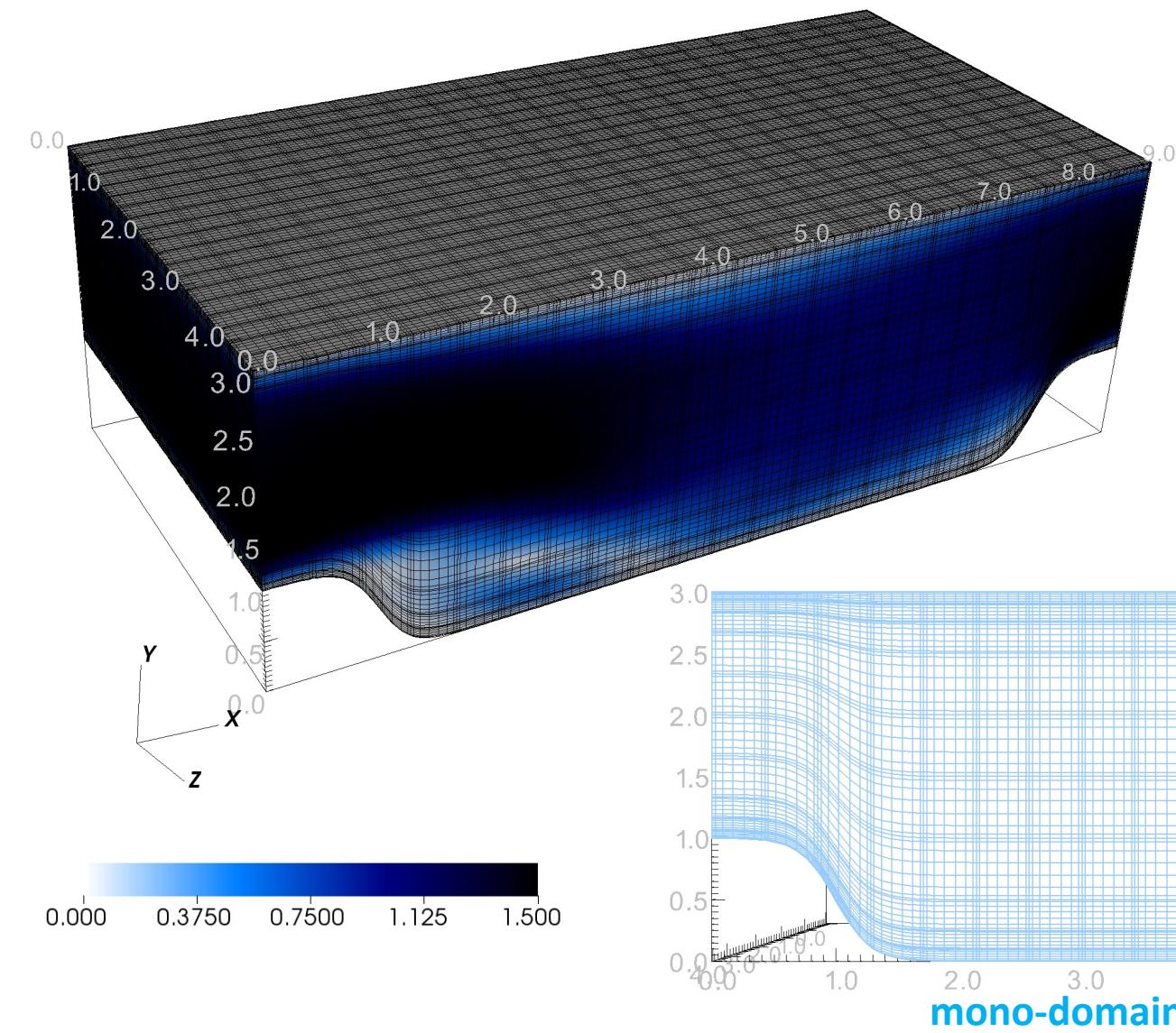
References/ reading material

- [1] MITTAL, K., DUTTA, S., & FISCHER, P. (2019). Nonconforming Schwarz-spectral element methods for incompressible flow. *Computers & Fluids*, 191, 104237.
- [2] MITTAL, K., DUTTA, S., & FISCHER, P. (2021). Multirate timestepping for the incompressible Navier-Stokes equations in overlapping grids. *Journal of Computational Physics*, 437, 110335.
- [3] MITTAL, K. (2019). Highly scalable solution of incompressible Navier-Stokes equations using the spectral element method with overlapping grids. *University of Illinois at Urbana-Champaign*.

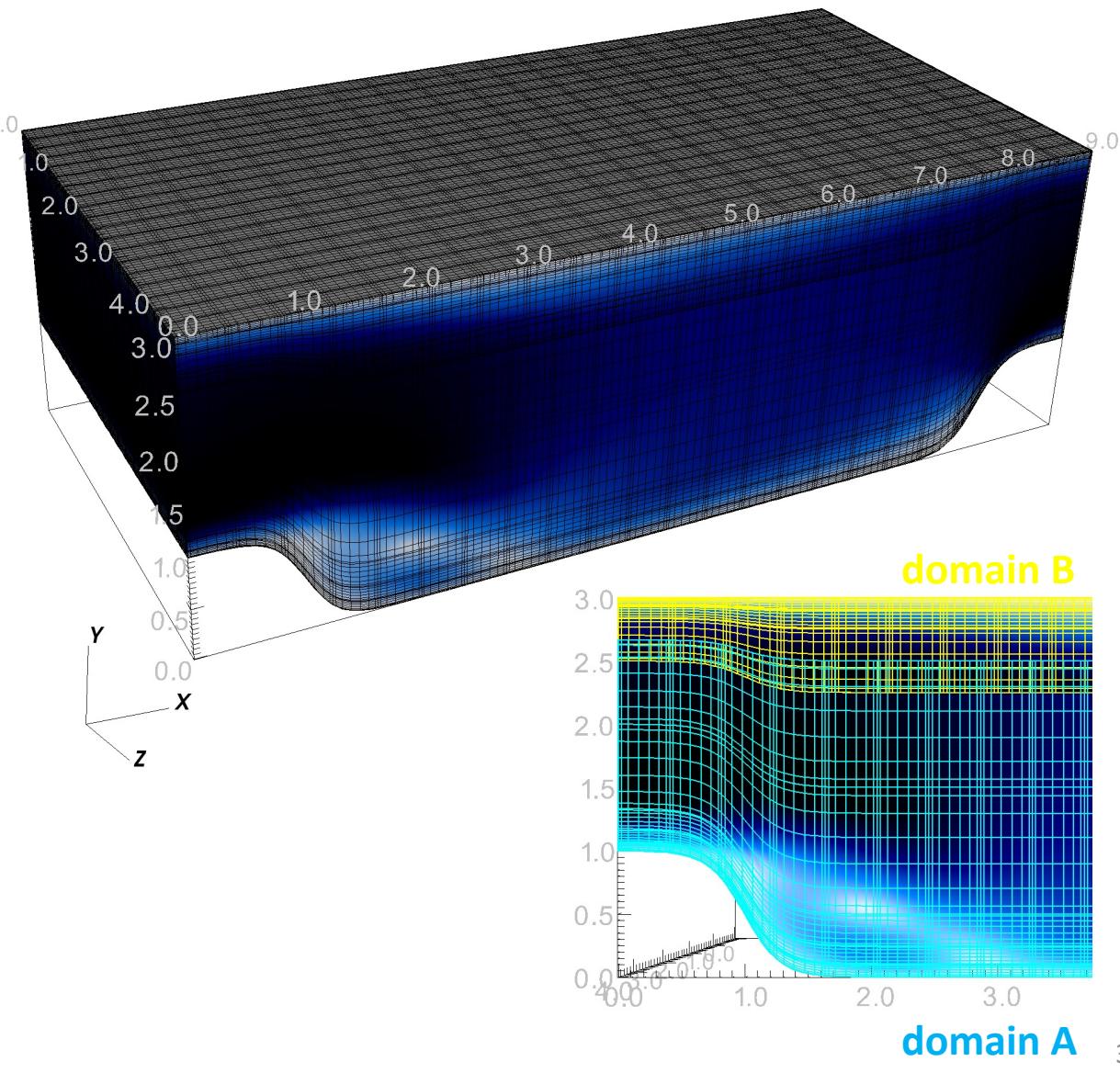
Today I will focus only on how to use NEKNEK.

Periodic hill example

NEK5000 – mono-domain simulation



NEK5000 - multiple domain simulation (NEKNEK)



DOS and DON'Ts



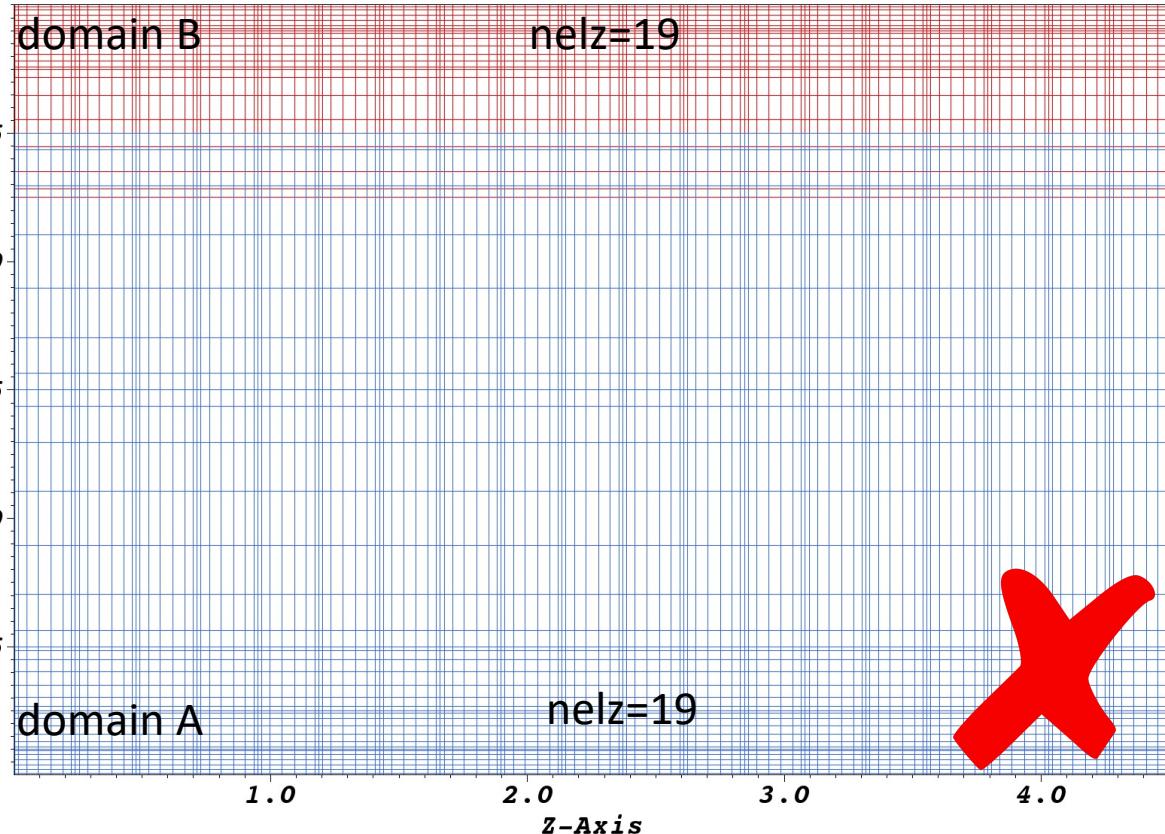
- Currently, applying a constant mass flux with param(54) and param(55) is not supported with overlapping overset grids.
(this is why we drive the flow using a constant acceleration term in userf)



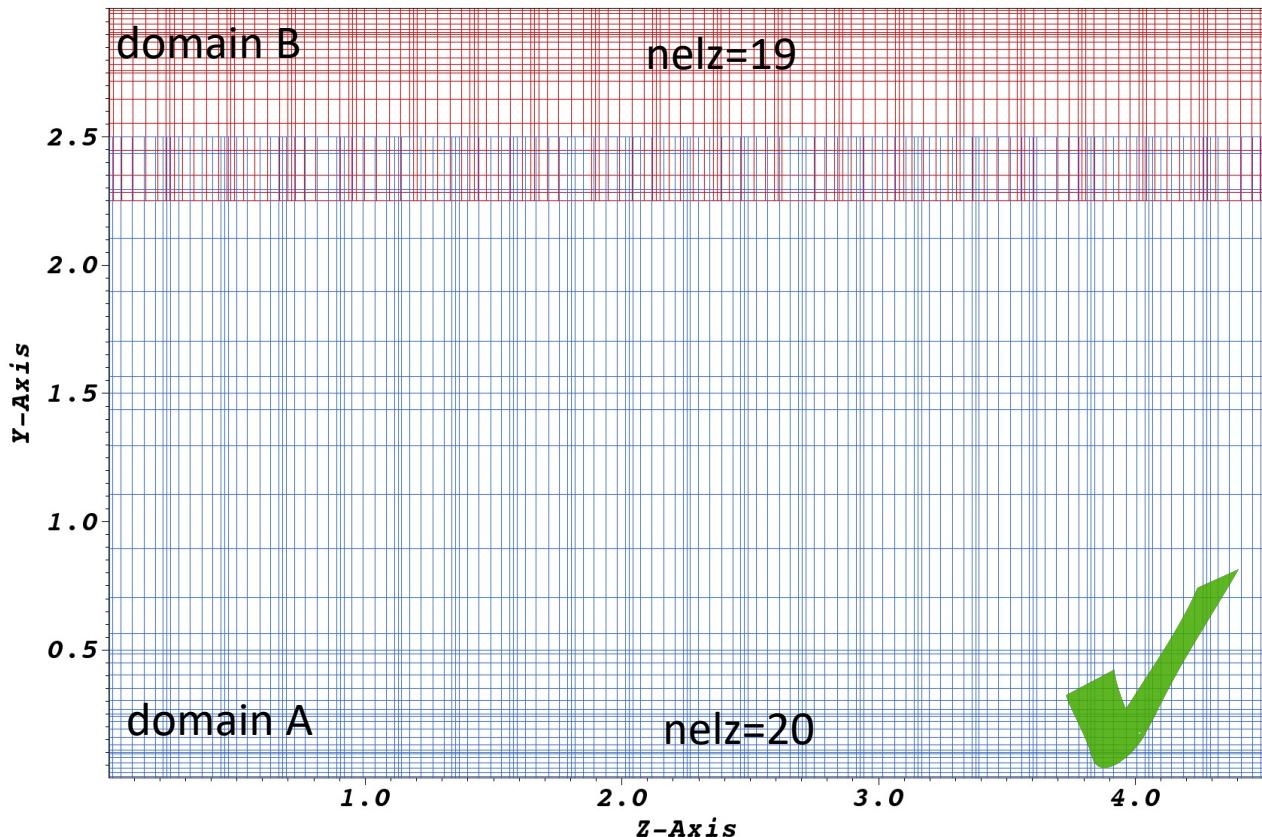
- multiple scalars
- RANS models
- moving geometries
- ...

DOs and DON'Ts

avoid aligned/exactly overlapped 1-1 GLL points
at the int interface



prefer not aligned GLL points
at the int interface



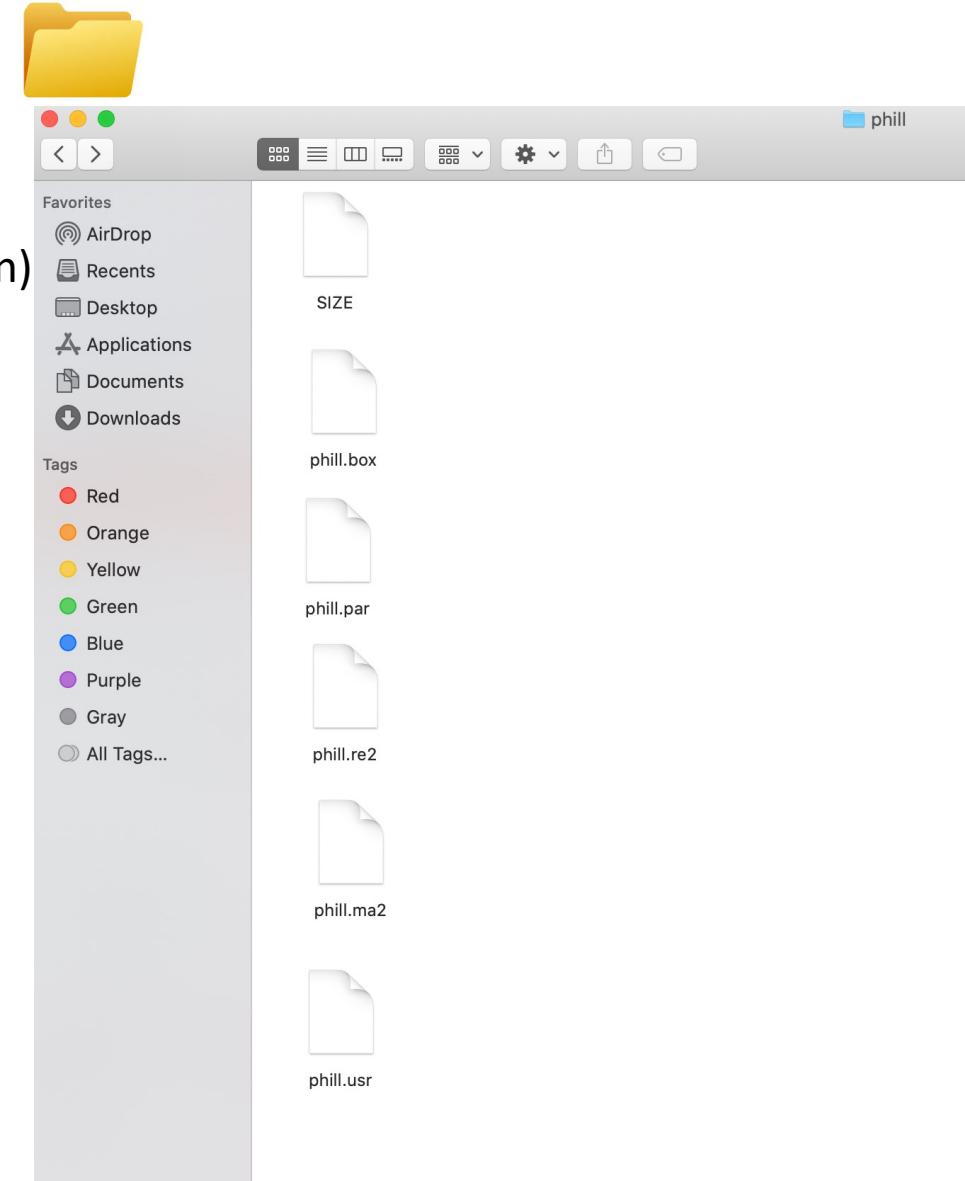
Workflow & files needed for a mono-domain/ single session NEK5000 simulation

files:

- 1 SIZE file
- 1 *box file (optional. used in case genbox is used for the grid generation)
- 1 *.rea (or *.par) file
- 1 *.map (or *.ma2) file
- 1 *.re2 (optional for large problems) file
- 1 *.usr file

steps:

- generate rea or re2 using genbox
 >genbox
 (...)
- generate map files using genmap
 >genmap
 (...)
- compile the code
 >makenek case_name(*.usr)
- run the code
 >nekmpi case_name np



Workflow & files needed for a 2-domain/ 2-session NEKNEK simulation

files:

- 1 SIZE file
- 2 *box files (optional. used in case genbox is used for the grid generation)
- 2 *.rea (or *.par) file
- 2 *.map (or *.ma2) file
- 2 *.re2 (optional for large problems) file
- 1 *.usr file

steps:

- generate rea or re2 using genbox

```
>genbox  
(...)1st domainA  
>genbox  
(...)2nd domainB
```

- generate map files using genmap

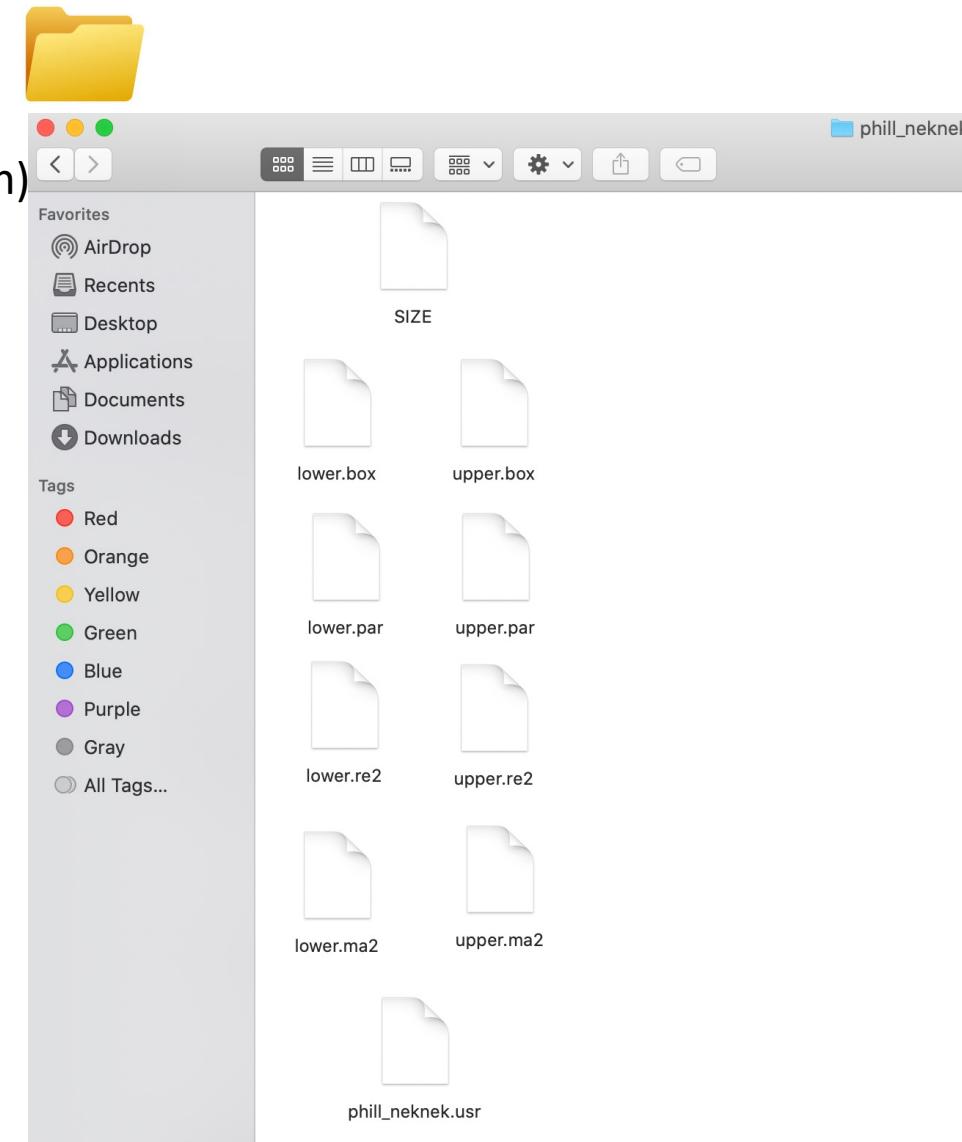
```
>genmap  
(...)1st domainA  
>genmap  
(...)2nd domainB
```

- compile the code

```
>makenek case_name(*.usr)
```

- run the code

```
>neknek domainA domainB np np
```



From a NEK5000 case to NEKNEK

The three key steps to running a case with NekNek are:

1. setting up the mesh with appropriate boundary conditions for the overlapping interface.
using **int** at the interfaces between the overlapping grids.

lower domain A:

```
#=====
#  
Box  
-22 5 -20  
0.0 9.0 1.0  
0.0 0.1 0.25 0.5 1.5 2.5  
0.0 4.5 1.0  
P ,P ,W ,int,P ,P  
nelx,nely for Box  
x0,xn,gain  
y0 y1 ratio  
z0,zn,gain
```

upper domain B:

```
#=====  
#  
Box  
-21 3 -19  
0.0 9.0 1.0  
2.25 2.75 2.9 3.0  
0.0 4.5 1.0  
P ,P ,int,W ,P ,P  
nelx,nely for Box  
x0,xn,gain  
y0 y1 ratio  
z0,zn,gain
```

2. specifying parameters to control stability and accuracy for the overlapping-Schwarz iterations.

defining the # of sub-iterations at each time-step **ngeom** and **ninter** in **usrdat()**.

```
subroutine usrdat() ! This routine to modify element vertices  
c implicit none  
  
include 'SIZE'  
include 'TOTAL'  
include 'NEKNEK' ! DKF: needed for neknek  
  
ngeom = 5 !DKF: 4 sub-iterations ($ngeom=5$) at each time-step  
ninter = 2 !DKF: the temporal extrapolation order ($ninter$) is 2.
```

3. modifying userbc to read Dirichlet boundary data for the overlapping interfaces.

typically using **valint(ix,iy,iz,ie,1)**, **valint(ix,iy,iz,ie,2)** etc in userbc()

```
c  
...  
  
if (imask(ix,iy,iz,ie).eq.0) then  
ux=0.0  
uy=0.0  
uz=0.0  
temp=0.0  
else  
ux = valint(ix,iy,iz,ie,1)  
uy = valint(ix,iy,iz,ie,2)  
uz = valint(ix,iy,iz,ie,3)  
if (nfld_neknek.gt.3) temp = valint(ix,iy,iz,ie,ldim+2)  
end if
```

From a NEK5000 case to NEKNEK

SIZE files:

mono-domain case:

```

c
c   Include file to dimension static arrays
c   and to set some hardwired run-time parameters
c
integer ldim,lx1,ldx,lx2,lx1m,lelg,lelt,lpmin,ldimt
integer lpelt,lbelt,toteq,lcvelt
integer lelx,lely,lelz,mxprev,lgmres,lorder,lhis
integer maxobj,lpert,nsessmax,lxo
integer lfdm,ldimt_proj,lelr

! BASIC
parameter (ldim=3)           ! domain dimension (2 or 3)
parameter (lx1=8)             ! GLL points per element along each direction
parameter (ldx=12)            ! GL points for over-integration (dealiasing)
parameter (lx2=lx1-2)          ! GLL points for pressure (lx1 or lx1-2)

parameter (lelg=13000)         ! max number of global elements
parameter (lpmin=32)           ! min number of MPI ranks
parameter (lelt=lelg/lpmin + 3) ! max number of local elements per MPI rank
parameter (ldimt=1)            ! max auxiliary fields (temperature + scalars)

! OPTIONAL
parameter (ldimt_proj=1)       ! max auxiliary fields residual projection
parameter (lelr=lelt)          ! max number of local elements per restart file
parameter (lhis=1)              ! max history/monitoring points
parameter (maxobj=2)            ! max number of objects
parameter (lpert=1)              ! max number of perturbations
parameter (toteq=1)              ! max number of conserved scalars in CMT
parameter (nsessmax=1)           ! max sessions to NEKNEK
parameter (lxo=lx1)              ! max GLL points on output (lxo>=lx1)
parameter (mxprev=20,lgmres=30)   ! max dim of projection & Krylov space
parameter (lorder=3)             ! max order in time
parameter (lx1m=1)                ! GLL points mesh solver
parameter (lfdm=0)                  ! unused
parameter (lelx=1,lely=1,lelz=1)    ! global tensor mesh dimensions

parameter (lbelt=1)              ! lelt for mhd
parameter (lpelt=1)              ! lelt for linear stability
parameter (lcvelt=1)              ! lelt for cvode

! INTERNALS
include 'SIZE.inc'

```

2-domain case:

```

c
c   Include file to dimension static arrays
c   and to set some hardwired run-time parameters
c
integer ldim,lx1,ldx,lx2,lx1m,lelg,lelt,lpmin,ldimt
integer lpelt,lbelt,toteq,lcvelt
integer lelx,lely,lelz,mxprev,lgmres,lorder,lhis
integer maxobj,lpert,nsessmax,lxo
integer lfdm,ldimt_proj,lelr

! BASIC
parameter (ldim=3)           ! domain dimension (2 or 3)
parameter (lx1=8)             ! GLL points per element along each direction
parameter (ldx=12)            ! GL points for over-integration (dealiasing)
parameter (lx2=lx1-2)          ! GLL points for pressure (lx1 or lx1-2)

parameter (lelg=13000)         ! max number of global elements
parameter (lpmin=32)           ! min number of MPI ranks
parameter (lelt=lelg/lpmin + 3) ! max number of local elements per MPI rank
parameter (ldimt=1)            ! max auxiliary fields (temperature + scalars)

! OPTIONAL
parameter (ldimt_proj=1)       ! max auxiliary fields residual projection
parameter (lelr=lelt)          ! max number of local elements per restart file
parameter (lhis=1)              ! max history/monitoring points
parameter (maxobj=2)            ! max number of objects
parameter (lpert=1)              ! max number of perturbations
parameter (toteq=1)              ! max number of conserved scalars in CMT
parameter (nsessmax=2)           ! max sessions to NEKNEK
parameter (lxo=lx1)              ! max GLL points on output (lxo>=lx1)
parameter (mxprev=20,lgmres=30)   ! max dim of projection & Krylov space
parameter (lorder=3)             ! max order in time
parameter (lx1m=1)                ! GLL points mesh solver
parameter (lfdm=0)                  ! unused
parameter (lelx=1,lely=1,lelz=1)    ! global tensor mesh dimensions

parameter (lbelt=1)              ! lelt for mhd
parameter (lpelt=1)              ! lelt for linear stability
parameter (lcvelt=1)              ! lelt for cvode

! INTERNALS
include 'SIZE.inc'

!DKF: 1 --> 2 for two domains

```

From a NEK5000 case to NEKNEK

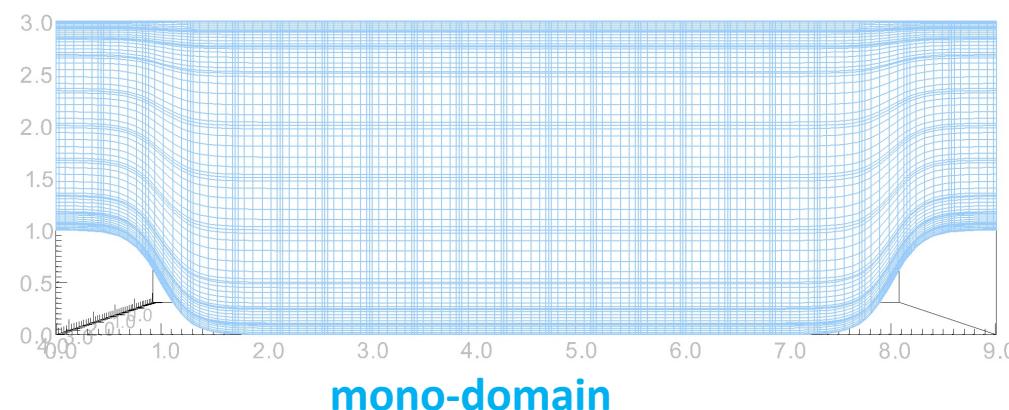
*.box files:

mono-domain case:

```

-3
1           spatial dimension
number of fields (U)
=====
# .box file for channel flow
#
# If nelx (y or z) < 0, then genbox automatically generates the
# grid spacing in the x (y or z) direction
# with a geometric ratio given by "ratio".
# ( ratio=1 implies uniform spacing )
#
# Note that the character bcs _must_ have 3 spaces.
#
=====
# Box
-21 10 -19          nelx,nely for Box
0.0 9.0 1.0          x0,xn,gain
0.0 0.1 0.25 0.5 1.0
1.5 2.0 2.5 2.75 2.9 3.0      y0 y1 ratio
0.0 4.5 1.0          z0,zn,gain
P ,P ,W ,W ,P ,P

```



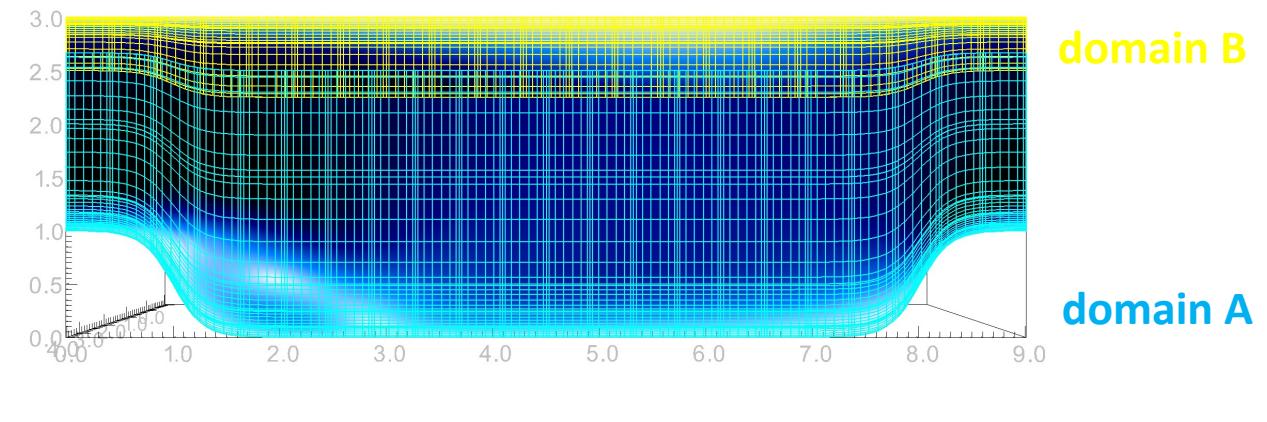
2-domain case:

domain A

```

-3
1           spatial dimension
number of fields (U)
=====
# .box file for channel flow
#
# If nelx (y or z) < 0, then genbox automatically generates the
# grid spacing in the x (y or z) direction
# with a geometric ratio ( ratio=1 implies uniform spacing )
#
# Note that the character bcs _must_ have 3 spaces.
#
=====
# Box
-22 5 -20          nelx,nely for Box
0.0 9.0 1.0          x0,xn,gain
0.0 0.1 0.25 0.5 1.5 2.5      y0 y1 ratio
0.0 4.5 1.0          z0,zn,gain
P ,P ,W ,int,P ,P

```



From a NEK5000 case to NEKNEK

*.par files:

mono-domain case:

```
#  
# nek parameter file  
#  
[GENERAL]  
#startFrom =  
stopAt = endTime #endTime numSteps  
endTime = 200.0  
numSteps =500  
  
dt = 5e-03  
timeStepper = bdf3 #char #steady  
variableDt = yes  
targetCFL = 0.4  
  
writeControl = runTime #runTime timeStep  
writeInterval = 0.1  
  
dealiasing = yes  
filtering = hpfrt #hpfrt, none, explicit  
filterWeight = 10.0  
filterCutoffRatio = 0.9  
  
[PROBLEMTYPE]  
equation = incompNS  
stressFormulation = no  
variableProperties = no  
  
[PRESSURE]  
preconditioner = semg_xxt  
residualTol = 1e-5  
residualProj = yes  
  
[VELOCITY]  
residualTol = 1e-8  
residualProj = yes  
density = 1.0  
viscosity = -100  
advection = yes
```

2-domain case:

domain A

```
#  
# nek parameter file  
#  
[GENERAL]  
#startFrom =  
stopAt = endTime #endTime numSteps  
endTime = 200.0  
numSteps =500  
  
dt = 5e-03  
timeStepper = bdf3 #char #steady  
variableDt = yes  
targetCFL = 0.4  
  
writeControl = runTime #runTime timeStep  
writeInterval = 0.1  
  
dealiasing = yes  
filtering = hpfrt #hpfrt, none, explicit  
filterWeight = 10.0  
filterCutoffRatio = 0.9  
  
[PROBLEMTYPE]  
equation = incompNS  
stressFormulation = no  
variableProperties = no  
  
[PRESSURE]  
preconditioner = semg_xxt  
residualTol = 1e-5  
residualProj = yes  
  
[VELOCITY]  
residualTol = 1e-8  
residualProj = yes  
density = 1.0  
viscosity = -100  
advection = yes
```

domain B

```
#  
# nek parameter file  
#  
[GENERAL]  
#startFrom =  
stopAt = endTime #endTime numSteps  
endTime = 200.0  
numSteps =500  
  
dt = 5e-03  
timeStepper = bdf3 #char #steady  
variableDt = yes  
targetCFL = 0.4  
  
writeControl = runTime #runTime timeStep  
writeInterval = 0.1  
  
dealiasing = yes  
filtering = hpfrt #hpfrt, none, explicit  
filterWeight = 10.0  
filterCutoffRatio = 0.9  
  
[PROBLEMTYPE]  
equation = incompNS  
stressFormulation = no  
variableProperties = no  
  
[PRESSURE]  
preconditioner = semg_xxt  
residualTol = 1e-5  
residualProj = yes  
  
[VELOCITY]  
residualTol = 1e-8  
residualProj = yes  
density = 1.0  
viscosity = -100  
advection = yes
```

From a NEK5000 case to NEKNEK

*.usr files (1/3, usrdat & usrdat2):

mono-domain case:

```
c-----  
c subroutine usrdat() ! This routine to modify element vertices  
  
c implicit none  
  
include 'SIZE'  
include 'TOTAL'  
  
return  
end  
  
c-----  
c subroutine usrdat2() ! This routine to modify mesh coordinates  
  
c implicit none  
  
include 'SIZE'  
include 'TOTAL'  
  
integer ntot  
real sa,sb,sc,xx,argx,A1  
  
ntot = nx1*ny1*nz1*nelt  
  
sa = 4.5  
sb = 3.5  
sc = 1./6  
  
do i=1,ntot  
  xx = xm1(i,1,1,1)  
  argx = sb*(abs(xx-sa)-sb)  
  A1 = sc + sc*tanh(argx)  
  ym1(i,1,1,1) = ym1(i,1,1,1) + (3-ym1(i,1,1,1))*A1  
enddo  
  
return  
end  
  
c-----
```

2-domain case:

```
c-----  
c subroutine usrdat() ! This routine to modify element vertices  
  
c implicit none  
  
include 'SIZE'  
include 'TOTAL'  
include 'NEKNEK' ! DKF: needed for neknek  
  
ngeom = 5 !DKF: 4 sub-iterations ($ngeom=5$) at each time-step.  
ninter = 2 !DKF: the temporal extrapolation order ($ninter$) is 2.  
  
return  
end  
  
c-----  
c subroutine usrdat2() ! This routine to modify mesh coordinates  
  
c implicit none  
  
include 'SIZE'  
include 'TOTAL'  
  
integer ntot  
real sa,sb,sc,xx,argx,A1  
  
ntot = nx1*ny1*nz1*nelt  
  
sa = 4.5  
sb = 3.5  
sc = 1./6  
  
do i=1,ntot  
  xx = xm1(i,1,1,1)  
  argx = sb*(abs(xx-sa)-sb)  
  A1 = sc + sc*tanh(argx)  
  ym1(i,1,1,1) = ym1(i,1,1,1) + (3-ym1(i,1,1,1))*A1  
enddo  
  
return  
end  
  
c-----
```

From a NEK5000 case to NEKNEK

domain A
domain B

*.usr files (2/3, userf):

mono-domain case:

```
c-----  
c subroutine userf(ix,iy,iz,eg) ! set acceleration term  
c  
c Note: this is an acceleration term, NOT a force!  
c Thus, ffx will subsequently be multiplied by rho(x,t).  
c  
c implicit none  
integer ix,iy,iz,eg  
  
include 'SIZE'  
include 'TOTAL'  
include 'NEKUSE'  
  
integer e  
e = gllel(eg)  
  
ffx = 0.052  
ffy = 0.0  
ffz = 0.0  
return  
end  
c-----
```

2-domain case:

```
c-----  
c subroutine userf(ix,iy,iz,eg) ! set acceleration term  
c  
c Note: this is an acceleration term, NOT a force!  
c Thus, ffx will subsequently be multiplied by rho(x,t).  
c  
c implicit none  
integer ix,iy,iz,eg  
  
include 'SIZE'  
include 'TOTAL'  
include 'NEKUSE'  
  
integer e  
! e = gllel(eg)  
  
ffx = 0.052  
ffy = 0.0  
ffz = 0.0  
return  
end  
c-----
```

no changes needed

From a NEK5000 case to NEKNEK

domain A
domain B

*.usr files (3/3, userbc & useric):

mono-domain case:

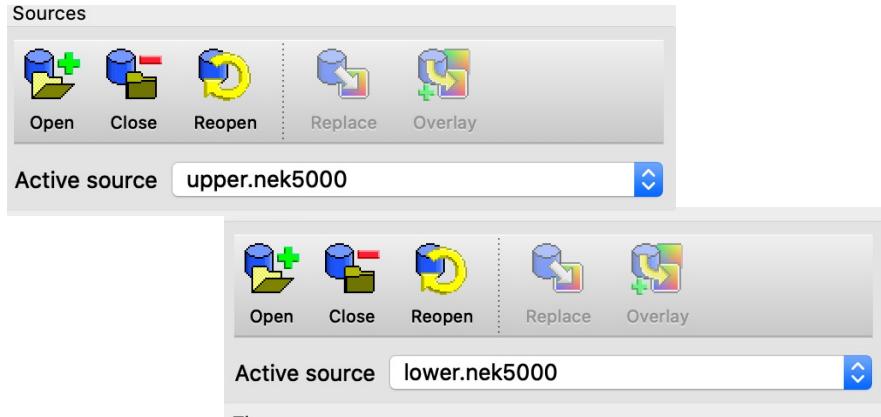
```
c-----  
c subroutine userbc(ix,iy,iz,iside,eg) ! set up boundary conditions  
c  
c NOTE ::: This subroutine MAY NOT be called by every process  
c  
c implicit none  
  
integer ix,iy,iz,iside,eg  
  
include 'SIZE'  
include 'TOTAL'  
include 'NEKUSE'  
  
ie = gglel(eg) ! global element number to processor-local element #  
ux=0.0  
uy=0.0  
uz=0.0  
temp=0.0  
  
return  
end  
c-----  
subroutine useric(ix,iy,iz,eg) ! set up initial conditions  
c  
c implicit none  
  
integer ix,iy,iz,eg  
  
include 'SIZE'  
include 'TOTAL'  
include 'NEKUSE'  
  
ux = 1.0  
uy = 0.0  
uz = 0.0  
temp = 0.0  
  
return  
end  
c-----
```

2-domain case:

```
c-----  
c subroutine userbc(ix,iy,iz,iside,eg) ! set up boundary conditions  
c  
c NOTE ::: This subroutine MAY NOT be called by every process  
c  
c implicit none  
  
integer ix,iy,iz,iside,eg  
  
include 'SIZE'  
include 'TOTAL'  
include 'NEKUSE'  
include 'NEKNEK' !DKF: needed for neknek  
  
ie = gglel(eg) ! global element number to processor-local element #  
!DKF: imask.eq.0 at points with  
  
if (imask(ix,iy,iz,ie).eq.0) then  
ux=0.0  
uy=0.0  
uz=0.0  
temp=0.0  
else  
ux = valint(ix,iy,iz,ie,1)  
uy = valint(ix,iy,iz,ie,2)  
uz = valint(ix,iy,iz,ie,3)  
if (nfld_neknek.gt.3) temp = valint(ix,iy,iz,ie,ldim+2)  
end if  
  
return  
end  
c-----  
subroutine useric(ix,iy,iz,eg) ! set up initial conditions  
c  
c implicit none  
  
integer ix,iy,iz,eg  
  
include 'SIZE'  
include 'TOTAL'  
include 'NEKUSE'  
  
ux = 1.0  
uy = 0.0  
uz = 0.0  
temp = 0.0  
  
return  
end  
c-----
```

Visualization using visit

- Load each domain's*.nek5000 file individually



- Create an “index database correlation”

