Nek5000 v19;
How to install and run a first case

Adam Peplinski
KTH Engineering Mechanics

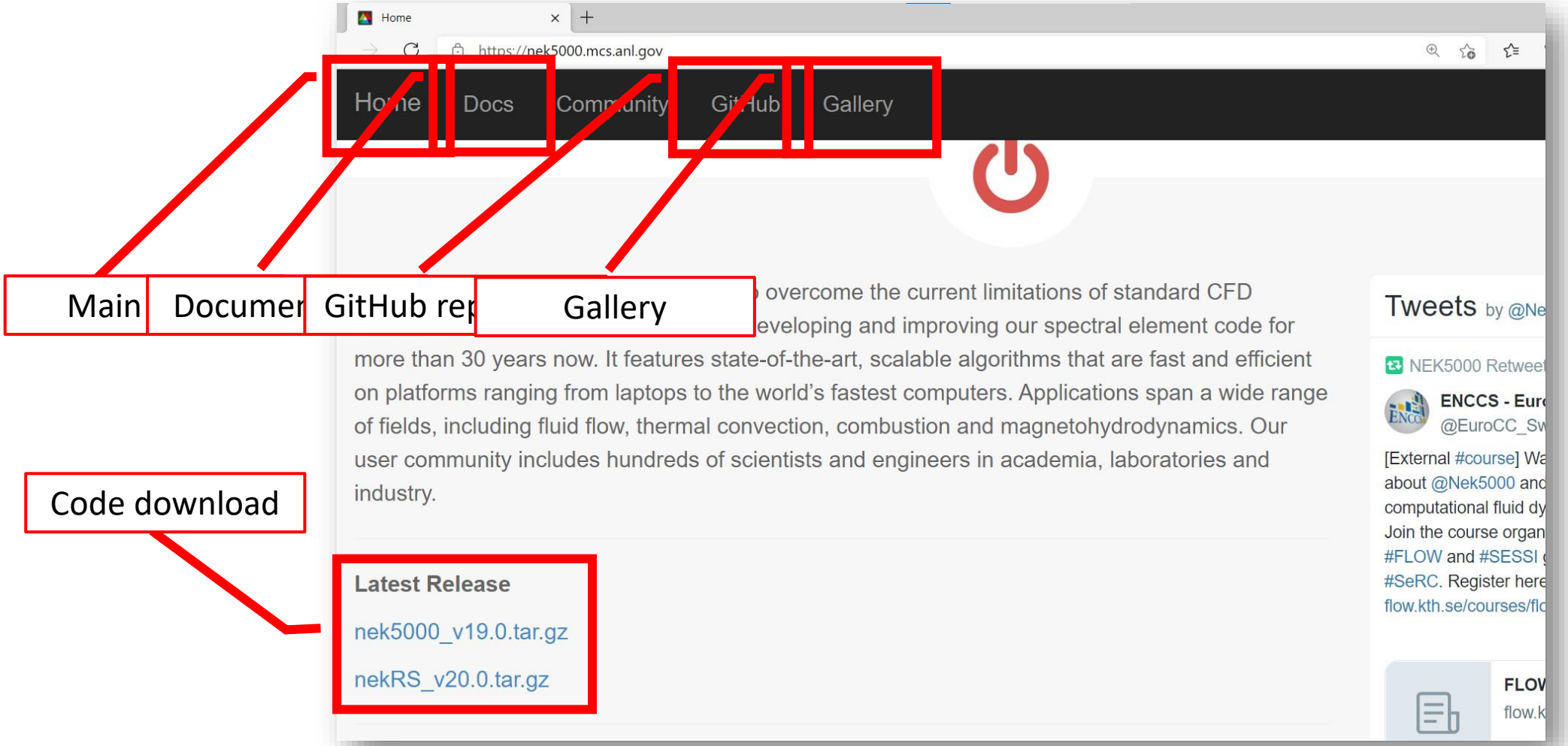
Based on material by:

Paul Fischer (UIUC), **Aleks Obabko** (ANL),
Philipp Schlatter (KTH), etc…

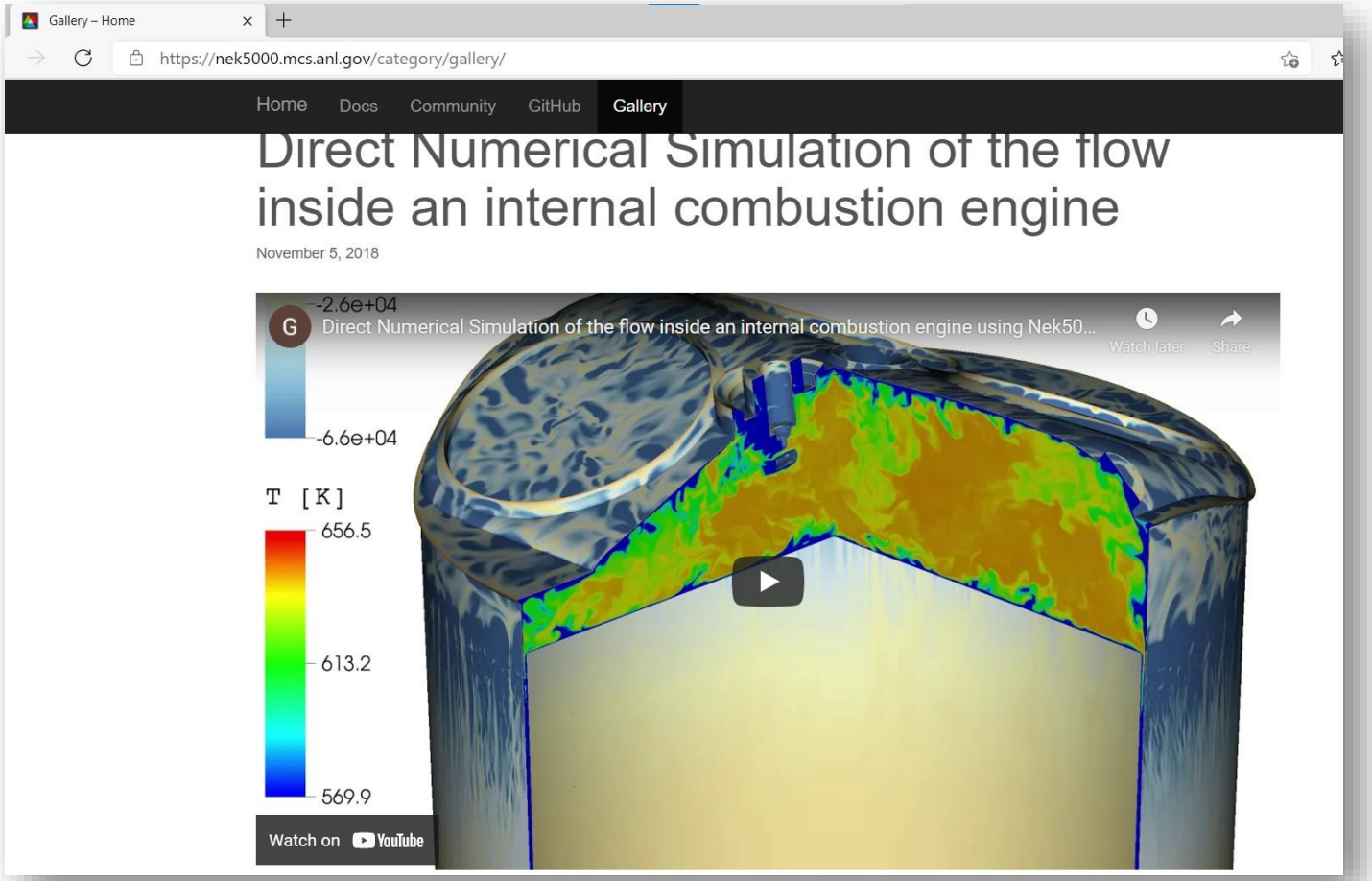
Outline

- Nek5000 resources
  - Documentation
  - Source code
- Code description
- Case setup
  - Compile, run, postprocess
  - Setting up a case
  - Meshing options

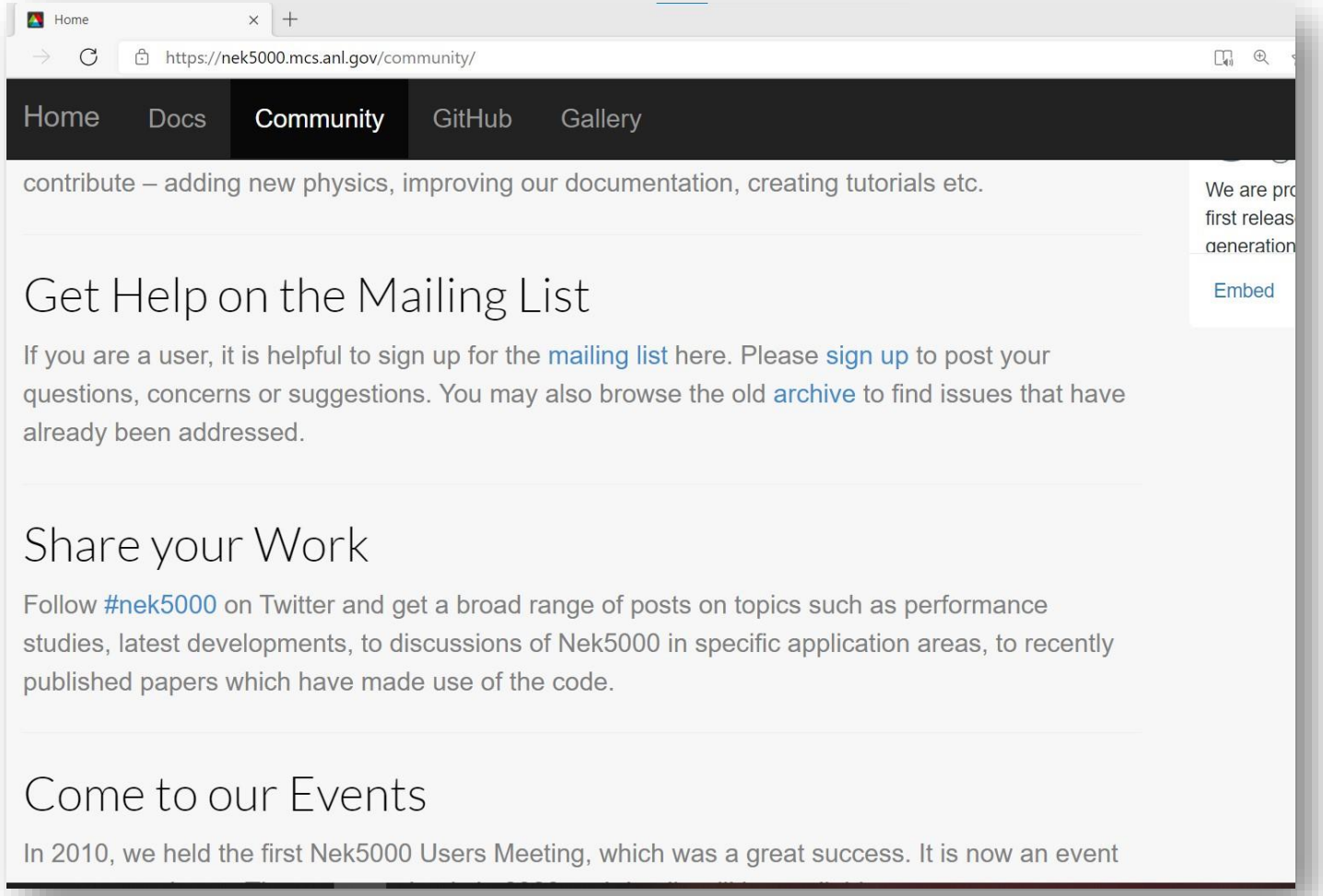
https://nek5000.mcs.anl.gov/



Gallery


Gallery – Home
https://nek5000.mcs.anl.gov/category/gallery/
Home
Docs
Community
GitHub
Gallery
Direct Numerical Simulation of the flow inside an internal combustion engine
November 5, 2018
Direct Numerical Simulation of the flow inside an internal combustion engine using Nek50...
Watch later
Share
-2.6e+04
-6.6e+04
T [K]
656.5
613.2
569.9
Watch on YouTube

Nek5000 google group



Documentation

Theory — Nek5000 17.0 docume

Not secure | nek5000.github.io/NekDoc/theory.html

Nek5000

17.0

Search docs

- Quickstart
- FAQ
- Tutorials
- Problem Setup
- Theory
- Appendices
- References

Docs » Theory

View page source

Theory

Computational Approach

The spatial discretization is based on the spectral element method (SEM) [Patera1984], which is a high-order weighted residual technique similar to the finite element method. In the SEM, the solution and data are represented in terms of N th-order tensor-product polynomials within each of E deformable hexahedral (brick) elements. Typical discretizations involve E=100–10,000 elements of order N=8–16 (corresponding to 512–4096 points per element). Vectorization and cache efficiency derive from the local lexicographical ordering within each macro-element and from the fact that the action of discrete operators, which nominally have $O(EN^6)$ nonzeros, can be evaluated in only $O(EN^4)$ work and $O(EN^3)$ storage through the use of tensor-product-sum factorization [Orszag1980]. The SEM exhibits very little numerical dispersion and dissipation, which can be important, for example, in stability calculations, for long time integrations, and for high Reynolds number flows. We refer to [Denville2002] for more details.

Nek5000 solves the unsteady incompressible two-dimensional, axisymmetric, or three-dimensional Stokes or Navier-Stokes equations with forced or natural convection heat transfer in both stationary (fixed) or time-dependent geometry. It also solves the compressible Navier-Stokes in the

QuickStart

Quickstart — Nek5000 17.0 docu

Not secure | nek5000.github.io/NekDoc/quickstart.html

Nek5000

17.0

Search docs

- Quickstart
  - Directory structure
  - Case files
  - Scripts
  - Running your very first simulation
  - Meshing
  - Visualization
- FAQ
- Tutorials
- Problem Setup
- Theory
- Appendices
- References

Docs » Quickstart

View page

Quickstart

Directory structure

Here's a brief description of each top-level directory:

/core

contains the Nek5000 application sources.

/bin

contains scripts for running nek5000 and manipulating its output.

/tools

contains the sources for the pre- and post-processing tools which are stand-alone.

/short-tests

contains light-weight regression tests for verification.

Tutorials

Tutorials — Nek5000 17.0 docu

Not secure | nek5000.github.io/NekDoc/tutorials.html

Nek5000

17.0

Search docs

Quickstart

FAQ

Tutorials

Periodic Hill

Conjugate Heat Transfer

Overlapping Overset Grids

Problem Setup

Theory

Appendices

References

Docs » Tutorials

View page so

Tutorials

The following tutorials are provided to familiarize beginners with the features and typical work of Nek5000. For a user brand new to Nek5000, we strongly recommend beginning with the Periodic Hill tutorial.

- Periodic Hill
- Conjugate Heat Transfer
- Overlapping Overset Grids

Previous

Nex

© Copyright 2015-2017, UCHICAGO ARGONNE, LLC.

Built with Sphinx using a theme provided by Read the Docs.

Tutorials

Periodic Hill — Nek5000 17.0 do

Not secure | nek5000.github.io/NekDoc/tutorials/perhill.html

Search docs

- Quickstart
- FAQ
- Tutorials
  - Periodic Hill
    - Pre-processing
    - Mesh generation
    - usr file
    - Control parameters
    - SIZE file
    - Compilation
    - Running the case
    - Post-processing the results
  - Conjugate Heat Transfer
  - Overlapping Overset Grids
- Problem Setup
- Theory
- Appendices
- References

```
    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

! apply mass flux to drive the flow such that Ubar = 1
param(54) = -1   ! x-direction
param(55) = 1    ! Ubar

return
end
```

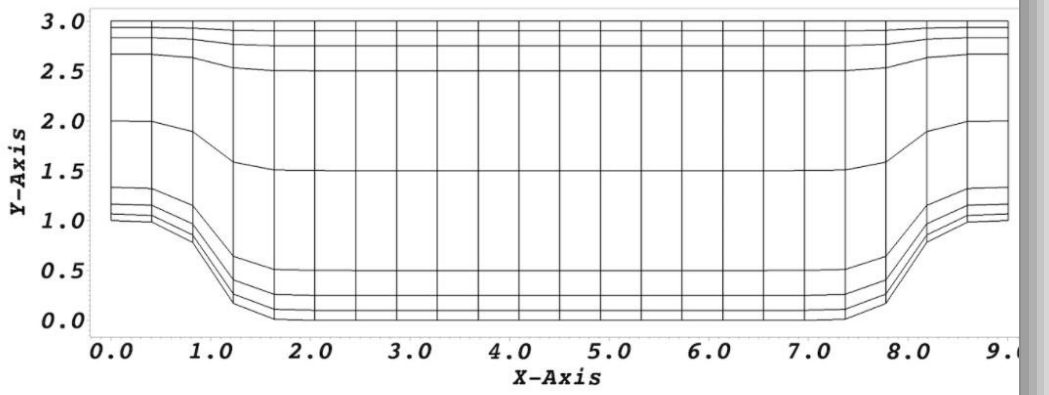


Fig. 1 Modified box mesh graded

Initial & boundary conditions

The next step is to specify the initial conditions. This can be done in the subroutine `useric` as

Tutorials

Overlapping Overset Grids — Ne

Not secure | nek5000.github.io/NekDoc/tutorials/neknek.html

Nek5000

17.0

Search docs

- Quickstart
- FAQ
- Tutorials
  - Periodic Hill
  - Conjugate Heat Transfer
  - Overlapping Overset Grids
    - Pre-processing
    - Mesh generation
    - usr file
    - Control parameters
    - SIZE file
    - Compilation
    - Running the case
    - Post-processing the results
- Problem Setup

```
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
```

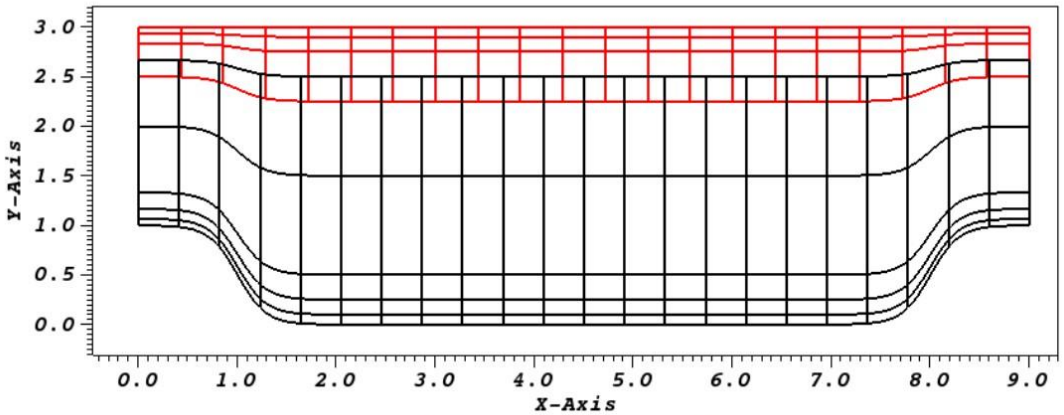


Fig. 5 Modified box mesh graded

Currently, applying a constant mass flux with `param(54)` and `param(55)` is **not** supported with overlapping overset grids. For this case, we drive the flow using a constant acceleration term in userf as:

Problem setup

Problem Setup — Nek5000 17.0

Not secure | nek5000.github.io/NekDoc/problem_setup.html

Nek5000
17.0

Search docs

Quickstart
FAQ
Tutorials
Problem Setup
Boundary Conditions
Case Files
Additional Features
Theory
Appendices
References

Problem Setup

A complete problem setup in Nek5000 requires a set of case files which are used for definin geometry and physics of a problem. Details of how to setup a case in Nek5000 are outlined following sections:

- Boundary Conditions
  - Fluid Velocity
  - Temperature and Passive Scalars
  - Internal Boundary Conditions
- Case Files
  - SESSION.NAME
  - Parameter File (.par)
  - Mesh File (.re2)
  - User Routines File (.usr)
  - SIZE
  - Mesh Partitioning File (.map/.ma2)
  - Restart/Output files (.f#####)
- Additional Features
  - History Points
  - Grid-to-Grid Interpolation
  - Averaging
  - Filtering
  - Objects

Appendices

Appendices — Nek5000 17.0 do

Not secure | nek5000.github.io/NekDoc/appendix.html

Nek5000

17.0

Search docs

Quickstart
FAQ
Tutorials
Problem Setup
Theory
Appendices
Build Options
Internal Input Parameters/Switches
Commonly used Variables
Commonly used Subroutines
Generating a Mesh with Genbox
Mesh Modification
Mesh Extrusion and Mirroring
References

Appendices

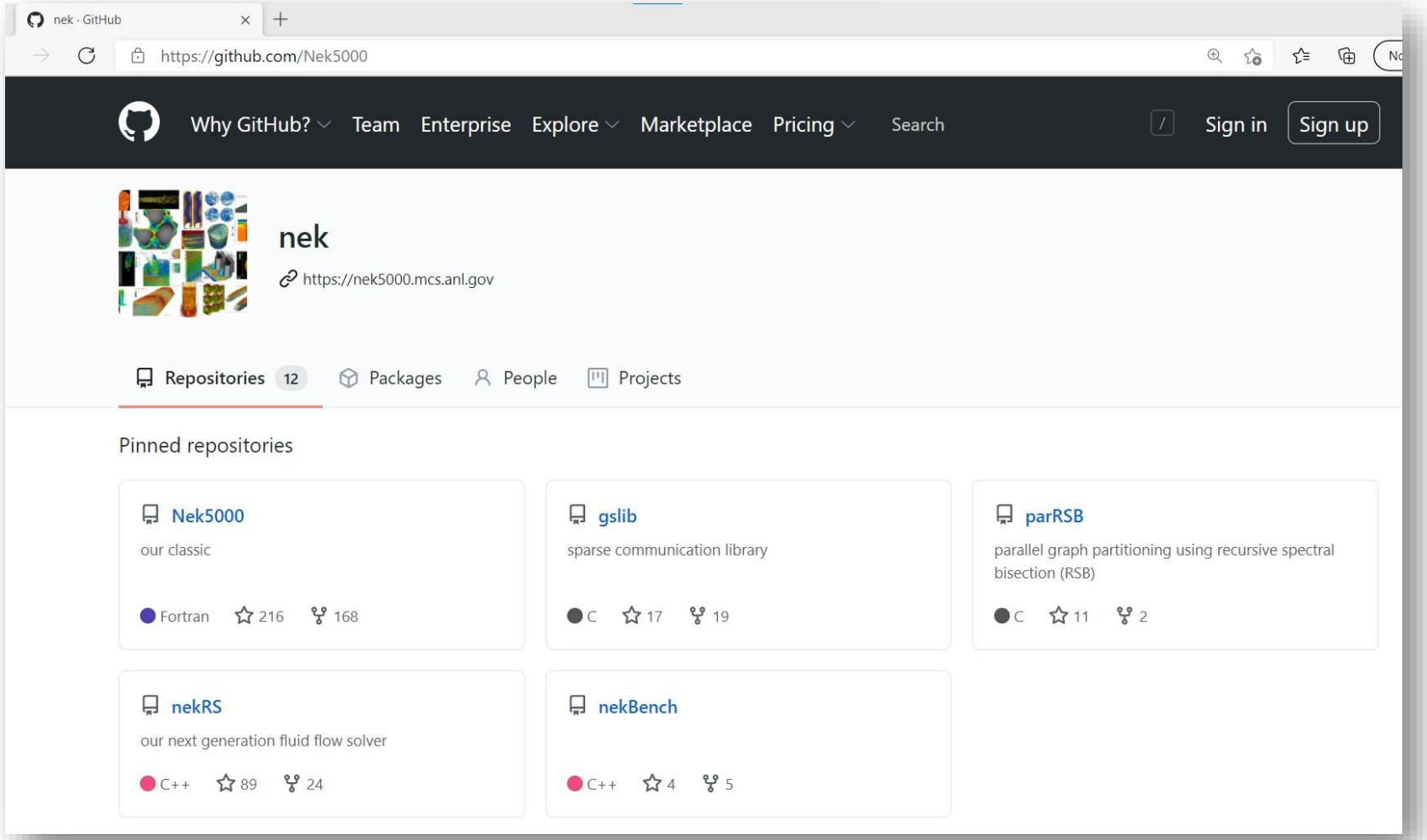
Build Options

The shell script `makenek` is designed to assist the compilation process of Nek5000. The scri
create a `makefile` based on the user settings section in `makenek`. The GNU gmake utility is
build Nek5000. Available configurations options:

Table 18 Compiler options¶

name	values	default	description
PPLIST	string		list of pre-processor symbols (CVODE, ...)
MPI	1, 0	1	use MPI (needed for a multiprocessor comp
FC	string	optional	Fortran compiler (mpif77)
CC	string	optional	C compiler (mpicc)
FCLAGS	string	optional	optional Fortan compilation flags
CCLAGS	string	optional	optional C compilation flags
SOURCE_ROOT	string	optional	path of Nek5000 source
USR	string	optional	object list of additional files to compile mak

GitHub repositories



GitHub repositories

Pinned repositories

Nek5000
our classic
Fortran 216 168

gslib
sparse communication library
C 17 19

parRSB
parallel graph partitioning using recursive spectral bisection (RSB)
C 11 2

nekRS
our next generation fluid flow solver
C++ 89 24

nekBench
C++ 4 5

Find a repository... Type Language

parRSB
parallel graph partitioning using recursive spectral bisection (RSB)
mesh partitioning

Top languages
C++ Fortran Python C
MATLAB

Nek5000 https://nek5000.mcs.anl.gov/

cfd hpc navier-stokes fluid high-order spectral flow anl sem

1,792 commits 5 branches 0 packages 2 releases

Branch: master New pull request

kent0 and stgeke write only on nio == 0 (#641)

3rd_party	update to parRSB 0.2
bin	Allow user to set NEK_SOURCE_ROOT (#622)
core	write only on nio == 0 (#641)
examples @ 35bac75	update to latest nekExamples
run	added AMG_DUMP to readme (#220)
short_tests	prefix AMG input files (#632)
tools	avoid arithmetic overflow for large element counts
.gitmodules	Add eddy_mv and chan2d short_test (#582)
.travis.yml	Import latest updates (#609)
LICENSE	update version info

GitHub repositories

Pinned repositories

Nek5000
our classic
Fortran 216 168

gslib
sparse communication library
C 17 19

parRSB
parallel graph partitioning using recursive spectral bisection (RSB)
C 11 2

nekRS
our next generation fluid flow solver
C++ 89 24

nekBench
C++ 4 5

Find a repository... Type Language

parRSB
parallel graph partitioning using recursive spectral bisection (RSB)
mesh partitioning

Top languages
C++ Fortran Python C
MATLAB

NekExamples

This repository contains number of examples illustrating different features of Nek5000 CFD solver. All exar separate sub-directories containing case description (README.md) and all the files necessary to run the si

Example overview

Some examples provide additional pdf files for a more complete description.

- `benard`
  - **Test name** : Benard_Ray9
  - **Purpose** : Rayleigh-Benard convection.
  - **Features** : * Customized acceleration term (userf)
- `blasius`
  - **Test name** : Blasius
  - **Purpose** : Blasius boundary layer test case in 2D.
  - **Features** :
    - "ON " boundary condition (Outflow, Normal only)
- `cone/cone016` , `cone/cone064` , `cone/cone256`
  - **Test name** : Cone16, Cone64, Cone256
  - **Purpose** : Convecting cone problem
  - **Features** :
- `conj_ht`
  - **Test name** : ConjHt
  - **Purpose** : 2D Poiseiulle flow moving between two heated finite thickness plates
  - **Features** : * Conjugate heat transfer * Scalar solve with CVODE
- `conj_ht_frz`

GitHub repositories

Pinned repositories

Nek5000
our classic
Fortran 216 168

gslib
sparse communication library
C 17 19

parRSB
parallel graph partitioning using recursive spectral bisection (RSB)
C 11 2

nekRS
our next generation fluid flow solver
C++ 89 24

nekBench
C++ 4 5

Find a repository... Type Language

parRSB
parallel graph partitioning using recursive spectral bisection (RSB)
mesh partitioning

Top languages
C++ Fortran Python C
MATLAB

GitHub - Nek5000/NekMatl
https://github.com/Nek5000/NekMatl

README.md

NekMatl

NekMetl has Nek5000 matlab codes for helping users to understand the essential components of Nek5000 based on the **spectral element method (SEM)**[1] and allowing users to experiment a new algorithmic implementation and its anaysis in an easier way using the matlab codes.

This includes demo codes:

- Spectral element discretizations (basic components)
- Spectral element multigrid Poisson solver: run `mgrid_demo.m`
- Spectral element Navier-Stokes solver: run `nst_demo.m`

More to be added (eg., projection, jacobian-free newton, etc)

The matlab codes are authored by Paul Fischer (fischerp@illinos.edu, fischer@mcs.anl.gov)

[1] M.O. Deville, P.F. Fischer, and E.H. Mund, High-Order Methods for Incompressible Fluid Flow, Cambridge University Press (2002).

Solver overview

Top-level directory:

- **/core:** contains the Nek5000 application sources.
- **/bin:** contains scripts for running nek5000 and manipulating its output.
- **/tools:** contains the sources for the pre- and post-processing tools which are stand-alone.
- **/short-tests:** contains light-weight regression tests for verification.
- **/run:** consistent place for users to place their problem cases.
- **/examples:** contains example problems. Note: NOT included in the master branch on the GitHub repo.
- **/3rd_party:** its purpose it to provide a consistent place for 3rd party code.

Scripts and tools

Useful batch scripts:

- **makenek <case>:** compiles your case
- **nek/nekb <case>:** runs a serial job in foreground or background
- **nekmpi/nekbmpi <case> <number of ranks>:** runs a parallel job
- **neknek <case1> <cas2> <ranks 1> <ranks 2>:** runs Nek5000 with two overlapping component grids
- **visnek <case>:** creates metadata file required by VisIt and ParaView.
- **mvn <old name> <new name>:** renames all case files
- **cpn <old name> <new name>:** copies all case files

Useful tools:

- Meshing
  - **genbox:** simple box type meshes
  - **nekmerge:** merging meshes
  - **prex/pretext:** more general mesh generator
  - **n2to3:** extrude 2D mesh in the 3rd dimension
  - **exo2nek:** exodus to nek5000 converter
  - **cgns2nek:** cgns to nek5000 converter
  - **gmsh2nek:** gmsh to nek5000 converter
- Setup file generator
  - **genmap:** generate global vertex numbering and element ordering
  - **gencon:** generate global vertex numbering
  - **reatore2:** split parameter and mesh information into 2 files

Compilation: makenek

- Bash script
- Collects variables
- Executes makenek.inc by calling nekconfig
- Executes make

```
: ${NEK_SOURCE_ROOT:="$HOME/Nek5000"}
echo "using source path: $NEK_SOURCE_ROOT"
VERMAKENEK="19.0.1"
set -a -e
$NEK_SOURCE_ROOT/bin/nekconfig $@
if [[ $1 != "clean" ]]; then
 $NEK_SOURCE_ROOT/bin/nekconfig -build-dep
 set -o pipefail
 make -j4 2>&1 | tee -a build.log
 set +o pipefail
fi
```

Compilation: makenek

name	values	default	description
PPLIST	string		list of pre-processor symbols (CVODE, ...)
MPI	1, 0	1	use MPI (needed for a multiprocessor computation)
FC	string	optional	Fortran compiler (mpif77)
CC	string	optional	C compiler (mpicc)
FCLAGS	string	optional	optional Fortan compilation flags
CCLAGS	string	optional	optional C compilation flags
SOURCE_ROOT	string	optional	path of Nek5000 source
USR	string	optional	object list of additional files to compile make intructions
USR_LFLAGS	string	optional	optional linking flags
PROFILING	1, 0	1	enable internal timers for performance statistics
VISIT	1, 0	0	Toggles Visit in situ. See Visit_in_situ for details
VISIT_INSTALL	string	VISIT in situ	Path to VISIT install path. See Visit_in_situ for details.
VISIT_STOP	true, false	false	When running VISIT in situ, simulation stops after step 1

PPLIST options

Symbol	Description
NOMPIIO	deactivate MPI-IO support
BGQ	use Blue Gene Q optimized mxm
XSMM	use libxsmm for mxm
CVODE	compile with CVODE support for scalars
VENDOR_BLAS	use VENDOR BLAS/LAPACK
EXTBAR	add underscore to exit call (for BGQ)
NEKNEK	activate overlapping mesh solver (experimental)
CMTNEK	activate discontinuous Galerkin compressible-flow solver (experimental)

makenek clean
makenek <case>

Compilation: makenek

name	values	default	description
PPLIST	string		list of pre-processor symbols (CVODE, ...)
MPI	1, 0	1	use MPI (needed for a multiprocessor computation)
FC	string	optional	Fortran compiler (mpif77)
CC	string	optional	C compiler (mpicc)
FCLAGS	string	optional	optional Fortan compilation flags
CCLAGS	string	optional	optional C compilation flags
USR	string	optional	object list of additional files to compile make intructions
USR_LFLAGS	string	optional	optional linking flags
PROFILING	1, 0	1	enable internal timers for performance statistics
VISIT	1, 0	0	Toggles Visit in situ. See Visit_in_situ for details
VISIT_INSTALL	string	VISIT in situ	Path to VISIT install path. See Visit_in_situ for details.
VISIT_STOP	true, false	false	When running VISIT in situ, simulation stops after step 1

```
${NEK_SOURCE_ROOT:=$(cd `dirname $0`/.. && pwd)
```

PPLIST options

Symbol	Description

```
"PPLIST config options:"
"   NOMPIIO        disable MPI-IO"
"   BGQ            use BGQ optimized mxm"
"   XSMM           use libxsmm for mxm"
"   CVODE          enable CVODE support"
"   HYPRE          enable HYPRE support"
"   VENDOR_BLAS    use VENDOR BLAS/LAPACK"
"   DPROCMAP       use distributed processor mapping"
"   PARRSB         use online RSB partitioner (experimental)
```

makenek clean

makenek <case>

Compilation: script

makefile_usr.inc

```
$(OBJDIR)/frame.o            :frame.f;                                $(F77) -c $(FL2) -I./ $< -o $@
$(OBJDIR)/mntrlog_block.o    :mntrlog_block.f MNTRLOGD;               $(F77) -c $(FL2) -I./ $< -o $@
$(OBJDIR)/mntrlog.o          :mntrlog.f MNTRLOGD;                     $(F77) -c $(FL2) -I./ $< -o $@
$(OBJDIR)/mntrtmr_block.o    :mntrtmr_block.f MNTRLOGD MNTRTMRD;      $(F77) -c $(FL2) -I./ $< -o $@
$(OBJDIR)/mntrtmr.o          :mntrtmr.f MNTRLOGD MNTRTMRD FRAMELP;    $(F77) -c $(FL2) -I./ $< -o $@
$(OBJDIR)/rprm_block.o       :rprm_block.f RPRMD;                     $(F77) -c $(FL2) -I./ $< -o $@
$(OBJDIR)/rprm.o             :rprm.f RPRMD FRAMELP;                   $(F77) -c $(FL2) -I./ $< -o $@
$(OBJDIR)/math_tools.o       :math_tools.f;                           $(F77) -c $(FL2) -I./ $< -o $@
$(OBJDIR)/spongebx.o         :spongebx.f SPONGEBXD;                   $(F77) -c $(FL2) -I./ $< -o $@
$(OBJDIR)/noisebx.o          :noisebx.f NOISEBXD;                     $(F77) -c $(FL2) -I./ $< -o $@
$(OBJDIR)/io_tools_block.o   :io_tools_block.f IOTOOLD;               $(F77) -c $(FL2) -I./ $< -o $@
$(OBJDIR)/io_tools.o         :io_tools.f IOTOOLD;                     $(F77) -c $(FL2) -I./ $< -o $@
$(OBJDIR)/chkpoint.o         :chkpoint.f CHKPOINTD;                   $(F77) -c $(FL2) -I./ $< -o $@
$(OBJDIR)/chkpt_mstp.o       :chkpt_mstp.f CHKPTMSTPD CHKPOINTD;      $(F77) -c $(FL2) -I./ $< -o $@
```

```
#parameters
export CASE="ext_cyl"
export NEK_SOURCE_ROOT="../../Nek5000"
export FC="mpifort"
export CC="mpicc"
export PPLIST=""
export USR="frame.o mntrlog_block.o mntrlog.o mntrtmr_bloc
USR+=" spongebx.o noisebx.o io_tools_block.o io_tools.o ch
export USR_LFLAGS=""

# arguments
args=("$@")
argsnr=$#

# check arguments
# parameters number check
if [ $[argsnr] -ne 1 ]; then
    error_quit 'Wrong arguments number!'
fi

for il in "$@"
do
case $il in
        --clean)
                ${NEK_SOURCE_ROOT}/bin/makenek clean
                shift
                ;;
        --all)
                ${NEK_SOURCE_ROOT}/bin/makenek ${CASE}
                shift
                ;;
        *)
                error_quit 'Wrong argument.'
                ;;
esac
done
```

Case files

- **SIZE**: contains some hardwired runtime parameters to dimension static arrays.
- **<case>.usr**: contains user specific code to initialize solver, set source terms and boundary conditions or to manipulate solver internals.
- **<case>. par**: contains runtime parameters.
- **<case>. re2**: contains mesh and boundary data.
- **<case>. ma2**: contains partioning data.
- **<case>. his**: contains probing points.
- **<case>. f00000**: contains checkpoint data.
- **<case>. nek5000**: contains metadata for VisIt or ParaView.
- **<case>. rea** (legacy): contains runtime parameters and mesh in ASCII. Replaced by .par and .re2 file.
- **<case>. map** (legacy): contains partioning data in ASCII.

SIZE

- Array sizes for static allocation
- Most often used variables

```
! BASIC
parameter (ldim=3)                ! domain dimension (2 or 3)
parameter (lx1=8)                 ! GLL points per element along each direction
parameter (lxd=12)                ! GL  points for over-integration (dealiasing)
parameter (lx2=lx1-0)             ! GLL points for pressure (lx1 or lx1-2)

parameter (lelg=100000)           ! max total number of elements
parameter (lpmin=1)               ! min MPI ranks
parameter (lpmax=1024)            ! max MPI ranks
parameter (ldimt=1)               ! max auxiliary fields (temperature + scalars)

! OPTIONAL
parameter (ldimt_proj=1)          ! max auxiliary fields residual projection
parameter (lhis=1000)             ! max history/monitoring points
parameter (maxobj=4)              ! max number of objects
parameter (lpert=1)               ! max number of perturbations
parameter (toteq=5)               ! 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=lx1)              ! GLL points mesh solver
parameter (lfdm=0)                ! set to 1 for fast diagonalization method
parameter (lelx=1,lely=1,lelz=1)  ! global tensor mesh dimensions

parameter (lelt=lelg/lpmin + 3)   ! max number of local elements per MPI rank
parameter (lbelt=1)               ! set to lelt for mhd
parameter (lpelt=1)               ! set to lelt for linear stability
parameter (lcvelt=lelt)           ! set to lelt for cvode

! INTERNALS
include 'SIZE.inc'
```

SIZE

- Array sizes for static allocation

```
! BASIC
parameter (ldim=3)              ! domain dimension (2 or 3)
parameter (lx1=8)               ! GLL points per element along each direction
parameter (lxd=12)              ! GL  points for over-integration (dealiasing)
parameter (lx2=lx1-0)           ! GLL points for pressure (lx1 or lx1-2)

parameter (lelg=100000)         ! max total number of elements
parameter (lpmin=1)             ! min MPI ranks
```

```
obj/bdry.o: In function `nekasgn_':
bdry.f:(.text+0x1e34): relocation truncated to fit: R_X86_64_PC32 against symbol `screv_' defined in COMMON section in obj/postpro.o
obj/coef.o: In function `glmapm3_':
coef.f:(.text+0x2418): relocation truncated to fit: R_X86_64_PC32 against symbol `screv_' defined in COMMON section in obj/postpro.o
coef.f:(.text+0x2437): relocation truncated to fit: R_X86_64_PC32 against symbol `screv_' defined in COMMON section in obj/postpro.o
coef.f:(.text+0x2456): relocation truncated to fit: R_X86_64_PC32 against symbol `screv_' defined in COMMON section in obj/postpro.o
coef.f:(.text+0x2561): relocation truncated to fit: R_X86_64_PC32 against symbol `screv_' defined in COMMON section in obj/postpro.o
obj/conduct.o: In function `fwght_':
conduct.f:(.text+0x553): relocation truncated to fit: R_X86_64_PC32 against symbol `scrdg_' defined in COMMON section in obj/convect.o
conduct.f:(.text+0x58e): relocation truncated to fit: R_X86_64_PC32 against symbol `scrdg_' defined in COMMON section in obj/convect.o
conduct.f:(.text+0x5b9): relocation truncated to fit: R_X86_64_PC32 against symbol `scrdg_' defined in COMMON section in obj/convect.o
obj/connect1.o: In function `setside_':
connect1.f:(.text+0xa8b): relocation truncated to fit: R_X86_64_PC32 against symbol `screv_' defined in COMMON section in obj/postpro.o
connect1.f:(.text+0xb5e): relocation truncated to fit: R_X86_64_PC32 against symbol `screv_' defined in COMMON section in obj/postpro.o
connect1.f:(.text+0xc38): additional relocation overflows omitted from the output
collect2: error: ld returned 1 exit status
makefile:119: recipe for target 'nek5000' failed
```

```
parameter (lelt=lelg/lpmin + 3)  ! max number of local elements per MPI rank
parameter (lbelt=1)              ! set to lelt for mhd
parameter (lpelt=1)              ! set to lelt for linear stability
parameter (lcvelt=lelt)          ! set to lelt for cvode

! INTERNALS
include 'SIZE.inc'
```

SIZE

- Array sizes for static allocation
- Most often used variables

```
! BASIC
parameter (ldim=3)              ! domain dimension (2 or 3)
parameter (lx1=8)               ! GLL points per element along each direction
parameter (lxd=12)              ! GL  points for over-integration (dealiasing)
parameter (lx2=lx1-0)           ! GLL points for pressure (lx1 or lx1-2)

parameter (lelg=100000)         ! max total number of elements
parameter (lpmin=1)             ! min MPI ranks
parameter (lpmax=1024)          ! max MPI ranks
parameter (ldimt=1)             ! max auxiliary fields (temperature + scalars)

! OPTIONAL
parameter (ldimt_proj=1)        ! max auxiliary fields residual projection
parameter (lhis=1000)           ! max history/monitoring points
parameter (maxobj=4)            ! max number of objects
parameter (lpert=1)             ! max number of perturbations
parameter (toteq=5)             ! 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=lx1)            ! GLL points mesh solver
parameter (lfdm=0)              ! set to 1 for fast diagonalization method
parameter (lelx=1,lely=1,lelz=1) ! global tensor mesh dimensions

parameter (lelt=lelg/lpmin + 3) ! max number of local elements per MPI rank
parameter (lbelt=1)             ! set to lelt for mhd
parameter (lpelt=1)             ! set to lelt for linear stability
parameter (lcvelt=lelt)         ! set to lelt for cvode

! INTERNALS
include 'SIZE.inc'
```

<case>.usr

User subroutines required by Nek5000

Mesh modifications

called after every step

```
subroutine uservp (ix,iy,iz,ieg)
subroutine usert  (ix,iy,iz,ieg)
subroutine userq  (ix,iy,iz,ieg)
subroutine userchk
subroutine userbc (ix,iy,iz,iside,eg)
subroutine useric (ix,iy,iz,ieg)
subroutine usrdat
subroutine usrdat2
subroutine usrdat3
```

userf

```
      subroutine userf  (ix,iy,iz,ieg)
      include 'SIZE'              ! NX1, NY1, NZ1, NELV, NID
      include 'NEKUSE'            ! FFX, FFY, FFZ
      include 'PARALLEL'          ! GLLEL
      include 'INPUT'             ! IF3D
      include 'SFD'               ! IFSFD, SFDCHI, BFS?c


      integer iel

      iel = gllel(ieg)

      SFD
      if (IFSFD) then
         FFX = FFX - SFDCHI*BFSX(ix,iy,iz,iel)
         FFY = FFY - SFDCHI*BFSY(ix,iy,iz,iel)
         if (IF3D) FFZ = FFZ - SFDCHI*BFSZ(ix,iy,iz,iel)
      else
         FFX = 0.0
         FFY = 0.0
         if (IF3D) FFZ = 0.0
      endif

      return
      end
```

Main memory structures stored in common blocks

Fortran include files

NEKUSE

```
c
c     Nekuse
c
      real x,y,z,r,theta,ux,uy,uz,un,u1,u2,trx,try,trz,trn,tr1,tr2,pa
     $    ,p0,ffx,ffy,ffz,temp,flux,hc,hrad,tinf,qvol,udiff,utrans
     $    ,si2,si3,sigma,turbk,turbe,ps(ldimt)
      common /nekuse/ x  ,y  ,z  ,r  ,theta
     $              , ux ,uy ,uz
     $              , un ,u1 ,u2
     $              , trx,try,trz
     $              , trn,tr1,tr2,pa,p0
     $              , ffx,ffy,ffz
     $              , temp,flux,hc,hrad,tinf,qvol
     $              , udiff,utrans
     $              , si2,si3,sigma
     $              , turbk,turbe
     $              , ps

      character*3     cbu
      common /nekusc/ cbu
```

SOLN

```
      parameter (lorder2 = max(1,lorder-2) )
c
c     Solution and data
c
      real bq(lx1,ly1,lz1,lelt,ldimt)
      common /bqcb/ bq

c     Can be used for post-processing runs (SIZE .gt. 10+3*LDIMT flds)
      real vxlag  (lx1,ly1,lz1,lelv,2)
     $    ,vylag  (lx1,ly1,lz1,lelv,2)
     $    ,vzlag  (lx1,ly1,lz1,lelv,2)
     $    ,tlag   (lx1,ly1,lz1,lelt,lorder-1,ldimt)
     $    ,vgradt1(lx1,ly1,lz1,lelt,ldimt)
     $    ,vgradt2(lx1,ly1,lz1,lelt,ldimt)
     $    ,abx1   (lx1,ly1,lz1,lelv)
     $    ,aby1   (lx1,ly1,lz1,lelv)
     $    ,abz1   (lx1,ly1,lz1,lelv)
     $    ,abx2   (lx1,ly1,lz1,lelv)
     $    ,aby2   (lx1,ly1,lz1,lelv)
     $    ,abz2   (lx1,ly1,lz1,lelv)
     $    ,vdiff_e(lx1,ly1,lz1,lelt)

c     Solution data
      real vx     (lx1,ly1,lz1,lelv)
     $    ,vy     (lx1,ly1,lz1,lelv)
     $    ,vz     (lx1,ly1,lz1,lelv)
```

Fortran include files

PARALLEL

```
      integer nelgt_max
      parameter(nelgt_max = 178956970)

      integer*8 nvtot
      integer nelg(0:ldimt1)
     $       ,lglel(lelt)
     $       ,gllel(lelg)
     $       ,gllnid(lelg)
     $       ,nelgv,nelgt
      common /hcglb/ nvtot,nelg,lglel,gllel,gllnid,nelgv,nelgt

      logical         ifgprnt
      common /diagl/  ifgprnt

      integer        wdsize,isize,lsize,csize,wdsizi
      common/precsn/ wdsize,isize,lsize,csize,wdsizi

!     crystal-router, gather-scatter, and xxt handles (xxt=csr grid solve)
!
      integer cr_h,gsh,gsh_fld(0:ldimt3),xxth(ldimt3)
      common /comm_handles/ cr_h,gsh,gsh_fld,xxth
```

GEOM

```
c     Geometry arrays
c
      real xm1(lx1,ly1,lz1,lelt)
     $    ,ym1(lx1,ly1,lz1,lelt)
     $    ,zm1(lx1,ly1,lz1,lelt)
     $    ,xm2(lx2,ly2,lz2,lelv)
     $    ,ym2(lx2,ly2,lz2,lelv)
     $    ,zm2(lx2,ly2,lz2,lelv)
      common /gxyz/ xm1,ym1,zm1,xm2,ym2,zm2

      real rxm1(lx1,ly1,lz1,lelt)
     $    ,sxm1(lx1,ly1,lz1,lelt)
     $    ,txm1(lx1,ly1,lz1,lelt)
     $    ,rym1(lx1,ly1,lz1,lelt)
     $    ,sym1(lx1,ly1,lz1,lelt)
     $    ,tym1(lx1,ly1,lz1,lelt)
     $    ,rzm1(lx1,ly1,lz1,lelt)
     $    ,szm1(lx1,ly1,lz1,lelt)
     $    ,tzm1(lx1,ly1,lz1,lelt)
     $    ,jacm1(lx1,ly1,lz1,lelt)
     $    ,jacmi(lx1*ly1*lz1,lelt)
      common /giso1/ rxm1,sxm1,txm1,rym1,sym1,tym1,rzm1,szm1,tzm1
     $              ,jacm1,jacmi
```

<case>.rea

```
****** PARAMETERS *****
  2.6099999      NEKTON VERSION
          2  DIMENSIONAL RUN
        122  PARAMETERS FOLLOW
  1.00000      p001 DENSITY
 -7500.00      p002 VISCOS
  0.00000      p003 : BETAG
  0.00000      p004 : GTHETA
  0.00000      p005 : PGRADX
  0.00000      p006
  1.00000      p007 RHOCP
  1.00000      p008 CONDUCT
  0.00000      p009
  0.00000      p010 FINTIME
  500003.      p011 NSTEPS
-7.000000E-04  p012 DT
  0.00000      p013 IOCOMM
  0.00000      p014 IOTIME
  500.000      p015 IOSTEP
  0.00000      p016 PSSOLVER: 0=default
  1.00000      p017
 0.500000E-01  p018 GRID < 0 --> # cells on screen
 -1.00000      p019 INTYPE
  7.00000      p020 NORDER
 0.100000E-09  p021 DIVERGENCE
 0.100000E-09  p022 HELMHOLTZ
  0.00000      p023 NPSCAL
 0.100000E-09
 0.100000E-04
  1.00000      p026 COURANT/NTAU
  3.00000      p027 TORDER
  0.00000      p028
  0.00000      p029
  0.00000      p030 > 0 ==> properties set in uservp()
  0.00000      p031 NPERT: #perturbation modes
  0.00000      p032 #BCs in re2 file, if > 0
  0.00000      p033 :
```

```
T       IFFLOW
F       IFHEAT
T       IFTRAN
T F F F F F F F F F F F  IFNAV & IFADVC (convection in P.S.
F F T T T T T T T T T T  IFTMSH (IF mesh for this field is
F       IFAXIS
F       IFSTRS
F       IFSPLIT
F       IFMGRID
F       IFMOVE
F       IFKEPS
F       IFMVBD
F       IFCHAR
  3.33333      3.33333      -0.833333      1.16667      XF
                                                  2nd line is
                                                  M,NELV
                                                  ****
                                                  *****
C Default
C Default
C Default
```

Restart options

Mesh settings

Real/integer parameters

<case>.par

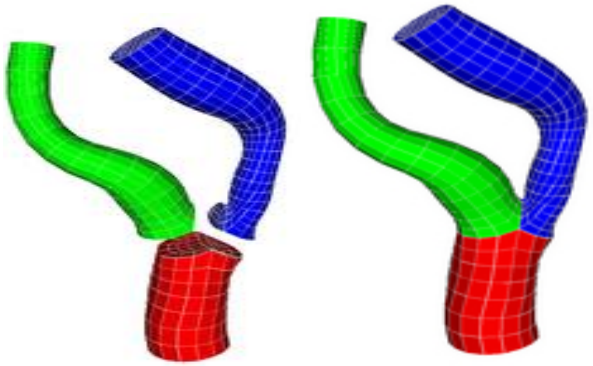
```
#
# nek parameter file
#

[SECTION]
key = value
...

[SECTION]
key = value
...
```

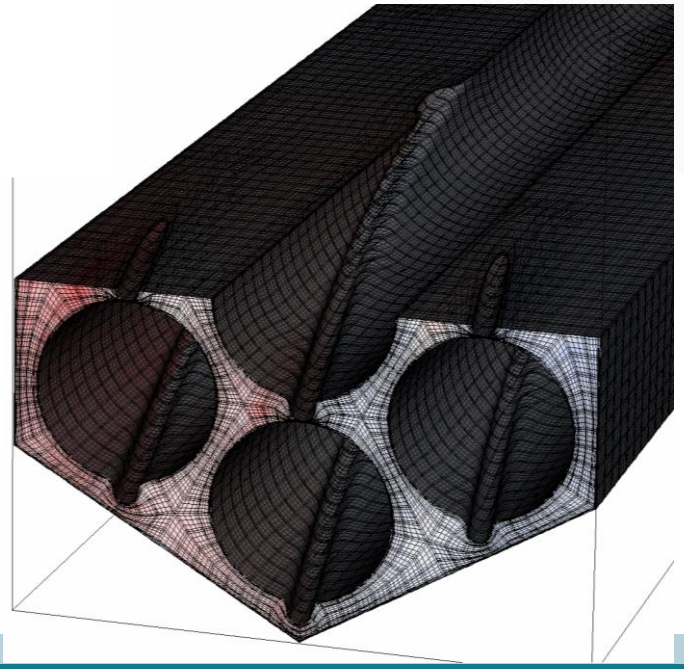
- GENERAL (mandatory)
- PROBLEMTYPE
- MESH
- VELOCITY
- PRESSURE (required for velocity)
- TEMPERATURE
- SCALAR%%
- CVODE

```
[GENERAL]
#startFrom = restart.fld
numSteps = 1000
dt = 1e-04
timeStepper = bdf3 #bdf2
writeInterval = 0

[PROBLEMTYPE]
equation = incompNS

[PRESSURE]
residualTol = 1e-08
residualProj = no

[VELOCITY]
residualTol = 1e-12
residualProj = no
density = 1
viscosity = -10
```

Some Meshing Options

- *genbox: unions of tensor-product boxes*
- *prenek: basically 2D + some 3D or 3D via extrusion (n2to3)*
- *Grow your own: 217 pin mesh via matlab; BioMesh*
- *3rd party: TrueGrid & conversion from ICEM (tet-to-hex),* ***gmsh****, CUBIT….*
  - *Existing convertors exo2nek,* ***gmsh2nek***
- *Morphing:*





Questions ?