NEKBONE KERNEL: A Single Core Kernel

Nekbone Kernel Release 2.0

April 11, 2013

Introduction to Nekbone_kernel

Nekbone_kernel is the single core kernel that exposes the main computational component of the mini-application, nekbone.

This kernel executes
$A \times u = w$
over a number of 3-dimensional elements by performing a series of matrix-matrix product evaluations.

More information about the nekbone kernel and nekbone can be found on the CESAR website:
cesar.mcs.anl.gov/content/software/thermal_hydraulics
or by contacting one of the developers.


```
Paul Fischer    :  fischer@mcs.anl.gov
Katherine Heisey:  heisey@mcs.anl.gov
```


This document contains the quick start guide followed by a more detailed explanation on modifying the nekbone kernel.

Running the Kernel

To run the kernel be sure to download the most recent version from `cesar.mcs.anl.gov/content/software/thermal_hydraulics` .
You must untar the nek_kernel-2.0.tgz, creating a kernel directory.
`tar -zxvf nek_kernel-2.0.tgz .`

This will create a nek_kernel-2.0directory with a `test/` and `src/` subdirectories. The `src/` directory contains the source code used to run a test. The `test` directory is where all tests should be ran.
`cd nek_kernel-2.0/test/`

A test is decribed by the SIZE file found in `nek_kernel-2.0/test/`. This file defines the polynomial order of the A matrix with parameters $lx1, ly1, lz1$. $lx1, ly1, lz1$ are the number of points in the x, y, z direction, respectfully, making the polynomial order of a simulation $lx1 - 1$. Thus, this is the complexity of the matrix-matrix product calculations done within the kernel. $lx1, ly1, lz1$ should all be equal for the 3-dimensional kernel calculation.

The SIZE file also defines the number of elements in the test case by the parameter, *lelt*. The higher *lelt* or $lx1$ is, the more computationally intense the test is.

To compile and link the code use the makenek script. The makenek script runs several checks on test environment and parameters before the source code is compiled and linked. It can all be done in one step:
`./makenek`

To run the kernel test:
`./nekkernel`

Clean-up `./makenek clean` will clean up the test directory, removing the .o files and the executable previously compiled. This will allow for a clean, recompile at the next `make` command.

Output The initialization time and the time spent in the `ax() subroutine` are printed to stdout.

makenek The makenek script provided in `nek_kernel-2.0/test/` allows the user to set compiler flags. Some of the commonly modified variables are explained below:

One of the important variables that is defined in the script is the source directory path, `SOURCE_ROOT=`. This should be set to the path to the source code. Since the tests are all ran from their own directory, this path can be locally defined as
`../src`
or more globally as the path from the user's HOME/ directory. As default, the path is set to
`$HOME/nek_kernel-2.0/src`
which assumes that the tarball was downloaded and unzipped in the HOME/ directory.

F77 is the compiler to be used. Nek_kernel has been tested with GNU's gfortran, PGI Portland, and INTEL serial compilers.

The G variable is for any compiler flags the user wants to include. A common setting is compiling with debugging turned on by setting G = "-g". For PGI Portland serial compilers, adding -Ktrap=fp will cause the test to exit when encountering any NaN values.

General optimization flags can be specified by setting the OPT_FLAGS_STD variable as desired. This will set the optimization level for a majority of the source files. If this is not specified, the code is compiled with $-O2$ and with $-O0$ when in debugging mode.

OPT_FLAGS_MAG is used to set the highest level of optimization, which is used on some of the of the more intricate files. If this variable is undefined, these files with be compiled with $-O3$ and $-O0$ when in debugging mode.

Initialization The initialization phase consists of:

- Finding the GLL points and weights
- Filling the u vector to be a random input vector

- Setting up the geometric factors used in the `ax() subroutine`

Note This is a kernel of the nekbone mini-application. It is the main component found in the mini-application, nekbone, and should only be ran on a single node. There is no communication, so no C compiler or MPI implementation is needed.

Nekbone and information regarding it can be found on the CESAR website:
`www.cesar.mcs.anl.gov/content/software/thermal_hydraulics/`