BeatBox User Workshop:
Hands-On Tutorial

June 25, 2013

Start

The computers should already be booted. Note that the operating system and all the files for this tutorial are on the memory stick, so this sticks should never be removed! If the screen goes locked due to long inactivity, your username is `beatbox` and your password is `beatbox`.

Notice the icon in the top left corner with the word BeatBox. Click at it and it will open a terminal (command line) window with the current directory `/beatbox`. This is our 'home' directory.

```
[beatbox@localhost ~]$ ls
bin         Desktop     Downloads                lost+found
CRN_model   Documents   FitzHugh-Nagumo_model    parallel_Hector
[beatbox@localhost ~]$ ls
```

We will work with scripts located in subdirectories `FizHugh-Nagumo_model`, `CRN_model` and `parallel_Hector`.

Sequential mode, FitzHugh-Nagumo model

Look into the FitzHugh-Nagumo model directory:

```
[beatbox@localhost ~]$ cd FitzHugh-Nagumo_model
[beatbox@localhost FitzHugh-Nagumo_model]$ ls *.bbs
```

(from this point on, we shall abbreviate the prompt in the screen listings to `[...]$`).

FitzHugh-Nagumo in 0D

First look at the contents of script `fhn0.bbs` using `more` command. This script was discussed yesterday, and it also contains detailed comments. Refresh in your memory what it is supposed to do. Run it:

```
[...]$ Beatbox_SEQ fhn0.bbs
```

This run will take a few seconds, while showing the progress of the phase trajectory on the screen. Make sure this has created file `fhn0.rec` using `ls -lt` command, and check its contents using `more` command.

FitzHugh-Nagumo in 1D

Now look at the next script, `fhn1.bbs` and run it:

```
[...]$ Beatbox_SEQ fhn1.bbs
```

This will show propagating pulses and will take a few seconds. In the end, the picture stops and the program hangs up until you press `Enter` key.

FitzHugh-Nagumo in 2D

```
[...]$ Beatbox_SEQ fhn2.bbs
```

This will show propagating rotating spiral wave and will take a minute or two. In the end, the picture will freeze for 3 seconds and then the program will terminate without having to press `Enter` key.

```
[...]$ gnuplot
        G N U P L O T
        ...
Terminal type set to 'wxt'
gnuplot> set size square
gnuplot> plot 'fhn2.trj' u 1:2 w l
```

This will show the trajectory. The command `set size square` here ensures that the scale of x and y coordinates will be the same, so a circle is drawn as a circle rather than an oval.

To finish, click on the terminal window, and

```
gnuplot> quit
[...]$
```

Now let us try to change the parameters of the last run. Launch the editor `Leafpad` through `Start - Accessories`, and open in it the file `/beatbox/FitzHugh-Nagumo_model/fhn.par`. Change

```
//FHN model kinetics parameters epsilon, beta, gamma
def real eps 0.30; def real bet 0.71; // rigid rotation, positive fi
//def real eps 0.20; def real bet 0.71;  // meander
//def real eps 0.3; def real bet 0.77; // negative filament tension
def real gam 0.50;
```

to

```
//FHN model kinetics parameters epsilon, beta, gamma
//def real eps 0.30; def real bet 0.71; // rigid rotation, positive
def real eps 0.20; def real bet 0.71;  // meander
//def real eps 0.3; def real bet 0.77; // negative filament tension
def real gam 0.50;
```

Run `fhn2.bbs` again, and plot the tip trajectory again. Instead of seeing just the new trajectory, you will see both the previous trajectory and the new trajectory.

Question. Explain why it happened. How this could have been avoided?

Answer. This happened because the new data were appended to the existing file `fhn2.trj` rather than overwriting it. As with `record` device, device `singz` has parameter `append` which defaults to 1. Hence to avoid appending new data to the old file, either of the two things could be done:

- Delete or rename the `fhn2.trj` file before the second run of `fhn2.bbs`.
- Add `append=0` to the body of `singz` device. In this case the old data in `fhn2.trj` will be overwritten with the new data.

If you need to run this script several times, the second solution may be preferable, since you would not need to remember to remove/rename the trj file every time.

Using File Manager, navigate into directory `/beatbox/FitzHigh-Nagumo_model/fhn2.dir`, open file `uvi0000000.png` using Image Viewer, and observe the high-resolution "movie" of the solution. Do the same for the `udg*.png` series.

Question. Explain why the first image in the `udg` series is so different from all subsequent images. How the script could be amended to avoid this.

Answer. The green component of the images represents the time derivative, which is found by substracting the u-field at the previous step from the current u-field. For the very first image, made at $t = 0$, there is no previous u-field, thus the time derivative is not found correctly. To avoid outputting the incorrect image file, one can amend the `fhn2.bbs` by changing the `when=` condition of the `k_imgout` device. This would have to be a new k-variable, which would equal zero at $t = 0$ and otherwise be identical to k-variable `out`. If we call the new variable `kout`, this may be done like this:

```
  ...
  def real begin;  // it is the very first step
  def real out;    // it is time to make outputs except for k_imgout
  def real kout; // it is time to make outputs for k_imgout
  def real dtime;  // time to make outputs or a step preceding that
  ...
k_func name=timing nowhere=1 pgm={ /* Define when to begin and end*/
  begin = eq(t,0); // beginning of simulation
  out   = eq(mod(t,tout),0); // time to make outputs every tout steps
  kout = out*gt(t,0);     // same as out except at t=0
  dtime = out + eq(mod(t,tout),tout-1); // when to call d_dt device
  ...
k_imgout when=kout
  ...
```

Exercise. Make both of the above amendments to the `singz` and `k_imgout` devices in the script `fhn2.bbs`, and run it again. Observe the difference in the fhn2.trj and solution output in the fhn2.dir directory.

Exercise (extra). Also, modify the script fhn2.bbs in such a way that the parameters `eps` and `bet` are given as the first and the second command-line arguments respectively. Check that everything works as expected by running

```
[...]$ Beatbox_SEQ fhn2.bbs 0.20 0.71
```

FitzHugh-Nagumo in 3D

Run `fhn3.bbs` in the background:

```
[...]$ Beatbox_SEQ fhn3.bbs > fhn3.out &
[1] 11039
[...]$
```

Here `> fhn3.out` redirects the standard output of the program to the file `fhn3.out`, and `&` in the end puts the execution into the "background", so you can continue working with the terminal window while BeatBox runs.

The full run will take quite a while, about half an hour, but some results could be seen already in a few minutes. The file `fhn3.out` will contain a copy of what you would have seen on screen in a normal, foreground run. The progress in terms of the created output files can be monitored like this:

```
[...]$ ls -lt fhn3.dir | head
total 1081600
-rw-rw-r--  1 beatbox  beatbox  1382989 23 Jun 09:28 000040.ppm
-rw-rw-r--  1 beatbox  beatbox        0 23 Jun 09:28 000041.ppm
-rw-rw-r--  1 beatbox  beatbox  1382989 23 Jun 09:28 000039.ppm
-rw-rw-r--  1 beatbox  beatbox  1382989 23 Jun 09:28 000038.ppm
-rw-rw-r--  1 beatbox  beatbox  1382989 23 Jun 09:28 000037.ppm
-rw-rw-r--  1 beatbox  beatbox  1382989 23 Jun 09:28 000036.ppm
-rw-rw-r--  1 beatbox  beatbox  1382989 23 Jun 09:28 000035.ppm
-rw-rw-r--  1 beatbox  beatbox  1382989 23 Jun 09:28 000034.ppm
-rw-rw-r--  1 beatbox  beatbox  1382989 23 Jun 09:28 000033.ppm
[...]$
```

These output files can be visualized using `ezview` (a visualizer based on graphical part of D. Barkley and M. Dowle's EZSCROLL) via the script `view.pl` located in this directory:

```
[...]$ ./view.pl fhn3.dir
```

There are two variations of the `fhn3.bbs` script in this directory (you do not need to run them!):

- `fhn3_PositiveTension.bbs` and
- `fhn3_NegativeTension.bbs`.

These are different from the base script `fhn3.bbs` in the following ways:

- The parameters are not taken from the file `fhn.par` but specified right within the script.
- The initial conditions are more sophisticated, using the phase distribution method.

Analyse the text of the three scripts and identify the places responsible for these differences. You can run these scripts, it will take about an hour each. The results can be viewed by

```
[...]$ ./view.pl fhn3_PositiveTension.dir
```

and

```
[...]$ ./view.pl fhn3_NegativeTension.dir
```

respectively.

FitzHugh-Nagumo in a 2D slice [1]

Run

```
[...]$ Beatbox_SEQ fhn_crossFieldStim_ffr_slice.bbs
```

This script will run a few seconds, and create a series of image files in `fhn_crossFieldStim_ffr_slice.dir` subdirectory. View these files using Image Viewer, as described previously. Observe that the simulations here are done in a complex geometry, this is a 2D slice of the rabbit ventricle geometry. The initial conditions are cross-field but that does not initiate a spiral wave due to geometric constraint.

Now run

[1] **Acknowledgement.** The realistic geometries used here in the 2D slice and 3D realistic anatomy simulations are by courtesy of Elizabeth M. Cherry and Flavio H. Fenton. If you use the anatomy geometries for any publications please cite as:
"The Rabbit Ventricles Geometry data comes from A.McCulloch, National Biomedical Computation Resource; NIH (USA) Grant P41 RR-08605 [F. J. Vetter, A. D. McCulloch, Three-dimensional analysis of regional cardiac function: a model of rabbit ventricular anatomy, Prog. Biophys. Mol. Biol. 69, 157-183, 1998]; The cartesian format: comes courtesy of Elizabeth M. Cherry and Flavio H. Fenton; The Beatbox format was produced by Ross McFarlane."

```
[...]$ Beatbox_SEQ fhn_spiral_ffr_slice.bbs
```

and view the files in `Beatbox_SEQ fhn_spiral_ffr_slice.dir` subdirectory. The initial conditions are now set using phase distribution method, which leads to initiation of a spiral wave at a location determined by k-variables `x0` and `y0` in the script.

Next, run

```
[...]$ Beatbox_SEQ fhn_spiral_ffr_slice_aniso.bbs
```

The main difference here from the previous case are the `anisotropy=1` and `normaliseVectors=1` parameters in the `state` command. Also, `diff` device now has `Dpar=D*2 Dtrans=D/2` (for diffusivities along and across the fibres, respectively) instead of `D=D` of the previous two examples: with anisotropy on, `diff` device no longer understands parameter `D` but requires `Dpar` and `Dtrans` instead. The spiral wave in this case is not stable and quickly drifts out of the tissue.

FitzHugh-Nagumo in a 3D ventricular geometry

There are four scripts on this topic:

```
[...]$ ls -l fhn_ffr*.bbs
-rwxrwxrwx  1 vadim  vadim  1980 23 Apr 09:29 fhn_ffr.bbs
-rwxrwxrwx  1 vadim  vadim  1936 23 Apr 09:27 fhn_ffr_iso.bbs
-rwxrwxrwx  1 vadim  vadim  2411 24 Apr 13:24 fhn_ffr_iso_xz.bbs
-rwxrwxrwx  1 vadim  vadim  2459 16 May 14:38 fhn_ffr_xz.bbs
[...]$
```

Here `iso` component of the name means isotropic diffusion, so the other two scripts are for anisotropic diffusion. The `xz` component of the name means phase-distribution initialization using x and z coordinates; the other two scripts are for cross-field initialization. Compare the contents of these scripts.

One of these examples has been pre-run: `fhn_ffr.bbs`, for anisotropic diffusion and cross-field stimulation. The results can be viewed with three different visualization methods, all exploiting `ezview` with different visualization parameters:

```
[...]$ ./view_ffr.pl fhn_ffr.dir/
```

```
[...]$ ./surface.pl fhn_ffr.dir/
```

```
[...]$ ./inside.pl fhn_ffr.dir/
```

The first and the second show evolution of the voltage on the surface of the tissue, the third shows propagation of the wavefront through the bulk of the tissue. The first method shows the surfaces solid (opaque), the second and the third show them semi-transparent. In all cases an attempt to visualize the singular filaments is done, but there are some false singular points detected. The detection of the filaments is controlled by line like

```
1                        show_filament
```

inside the `*.pl` file; changing here 1 to 0 will abolish the attempts to visualize the singular filaments.

`ezview` is an interactive program: scrolling through input files can be paused by pressing `P` with the keyboard focus in the graphics window; then the image can be rotated using arrows or mouse click-and-drag.

Sequential mode, Courtemanche et al. 1998 (CRN) model

CRN model in 0D

```
[beatbox@localhost FitzHugh-Nagmo_model]$ cd
[beatbox@localhost ~]$ cd ~/CRN_model
[beatbox@localhost CRN_model]$ ls *.bbs
pd_crn0.bbs     pd_crn1.bbs     pd_crn2.bbs
[beatbox@localhost CRN_model]$ Beatbox_SEQ pd_crn0.bbs
```

The script will produce a series of action potentials, draw them in two different forms and write them to disk files.

Inspect this script. Follow the logic of the control devices and of the feedback-driven stimulation protocol. It is similar to `fhn0.bbs` considered earlier.

Guess what is the function of device `k_print` (it was not explained in the lecture yesterday). Check your guess by looking at the contents of the file `pb_crn0.vtg` using `more` command. Plot the action potentials using `gnuplot`.

Compare the functions of device `k_print` and `record`. Plot the action potential using `gnuplot`:

```
[...]$ gnuplot
        G N U P L O T
        ...
Terminal type set to 'wxt'
gnuplot> plot 'pd_crn0.rec' u 0:1 w l
gnuplot> plot 'pd_crn0.rec' u ($0*0.005):1 w l
```

In the first case, the plot was of the voltage against the line number of the file `pd_crn0.rec`. In the second case, it was against the time in milliseconds (why? find 0.005 in the file `pd_crn0.bbs` for a hint).

Plot the contents of `pd_crn0.vtg` by `gnuplot` again and look closer at the graph. The shape of the action potentials (APs) is not constant: during the first few APs, some tendency to alternances can be noticed; however subsequently it all settles down to a stable AP profile.

Observe usage of string macros `win1` and `win2`. Figure out how the action potentials in the right half of the graphics window are drawn. Why `lines=0.1` and what will happen if it is set to 0? to 1?

CRN model in 1D

Once `pd_crn0.bbs` has been run, we should have the record file `pd_crn0.rec`, so we can now do the next step:

```
[...]$ Beatbox_SEQ pd_crn1.bbs
```

Inspect the script `pd_crn1.bbs`. Similar to `fhn1.bbs` considered earlier, the script `pd_crn1.bbs` uses the record file `pd_crn0.rec` to set non-homogeneous Dirichlet boundary conditions on the left end of the interval, to initiate propagating waves.

The script `pd_crn1.bbs` produces files `pd_crn1.rec` and `pd_crn1.vtg`. You can visualize them using `gnuplot` as before (look again at the appropriate instructions for `fhn1.bbs` and modify them for the present case appropriately). The first one contains record of all variables, not only voltage. This is important for using it for initial conditions for the next script, `pd_crn2.bbs`.

Question. Find out, where in the script `pd_crn1.bbs` it is determined how long does the simulation run. Currently it runs for as long as it takes for *the second* pulse to reach the middle of the interval, so the output file `crn1.vtg` contains one full AP record plus a bit. How to modify the script so it stops when *the third* pulse reaches the middle of the interval?

Answer. You need to change the line

```
def int countmax 1;     // when to stop
```

to

```
def int countmax 2;     // when to stop
```

Make sure you understand how it works, and ask the demonstrators if you don't.

Exercise. Make this modification of the k-variable `countmax` in `pd_crn1.bbs` and run the resulting script (**NB** don't forget to check whether the file `crn1.vtg` is going to be overwritten or appended!). Check the result visually during the run, and also by visualizing `crn1.vtg` afterwards. Visualize also the AP as recorded in `crn1.rec`. There is only one AP there — explain why. Ask the demonstrators if you can't.

CRN model in 2D

This is a relatively long run, so it does not contain run-time graphics and can be run in the background:

```
[...]$ Beatbox_SEQ pd_crn2.bbs > pd_crn2.out &
```

Its progress can be then monitored by checking the contents of the image directory,

```
[...]$ ls -lt pd_crn2.dir
total 216
-rw-r--r--  1 vadim  vadim  26604 23 Jun 14:13 uvi0000300.png
-rw-r--r--  1 vadim  vadim  25875 23 Jun 14:13 uvi0000200.png
-rw-r--r--  1 vadim  vadim  25157 23 Jun 14:12 uvi0000100.png
-rw-r--r--  1 vadim  vadim  23444 23 Jun 14:12 uvi0000000.png
546 14:13:38 CRN_model$
```

or by plotting the tip trajectory:

```
[...]$ gnuplot
        G N U P L O T
        ...
Terminal type set to 'wxt'
gnuplot> plot 'pd_crn2.trj' u 1:2 w lp
```

The parameter `w lp` here stands for `with linespoints`: in this model, the movement of the tip is not continuous, and marking every position of the tip with a symbol helps to identify the jumps of the trajectory.

By repeating the `plot` command while still within `gnuplot` from time to time (use arrow key to return the previous command on the command line), you can monitor the progress of the tip.

You can also visualize the image files in `pd_crn2.dir` with Image Viewer as before.

One rotation of a spiral may take about XXXX of minutes in sequential run.

Parallel mode, remote runs on HECToR

```
[beatbox@localhost CRN_model]$ cd
[beatbox@localhost ~]$ cd parallel_Hector
[beatbox@localhost parallel_Hector]$ ls *.bbs
fhn3_NegativeTension.bbs      fhn_ffr_xz.bbs fhn_ffr_xz_scaling.bbs
[beatbox@localhost parallel_Hector]$ ls *.bbg
ffr.bbg
[beatbox@localhost parallel_Hector]$ ls *.rec
fhn1_NegativeTension.rec      fhn1_PositiveTension.rec
[beatbox@localhost parallel_Hector]$ ls *.pbs
fhn3_NegativeTension.pbs      fhn_ffr_xz.pbs
```

This part of the tutorial is about running jobs on a remote supercomputer. For this you need a username and password. A number of guest accounts have been created on HECToR for this workshop which will be given to you on a separate piece of paper at your terminal. In this manual, we will designate these as `<username>` and `<password>`.

Advance apology: the dedicated accounts for this workshop are only made available on the day of the hands-on tutorial. Hence we are unable to test everything beforehand, and our instructions are partly based on guesswork. In real life, some of the instruction may have to be adjusted to the circumstances. We are sorry for possible inconveniences.

The overall schedule of our exercise will be:

- Prepare the jobs.
- Copy all necessary files to the remote computer.
- Submit the jobs on the remote computer.
- Copy the results back to the local computer.

Preparing the jobs.

The jobs on HECToR are defined by submission scripts, which are files with extention `pbs`. Syntactically, they are `bash` scripts with some special sort of comments. Here is the contents of one of them:

```
520 15:35:04 parallel_Hector$ more fhn_ffr_xz.pbs
#!/bin/bash --login
#PBS -N FhnFfrXz
#PBS -l mppwidth=96
#PBS -l mppnppn=32

# time requested
#PBS -l walltime=00:10:00
#PBS -A e203

# change to the directory where you will run the simulation.
cd /home/e203/e203/ivb203/work/FitzHughNagumo_model

ulimit -s unlimited

# Launch the parallel job
aprun -n 96 -N 32 ./Beatbox fhn_ffr_xz.bbs -profile -verbose -nograph
```

The special comments are those starting with `#PBS`.

The first of them, `#PBS -N FhnFfrXz` says that the name of the job, by which it can be recognized in the queue, is `FhnFfrXz`.

The comment `#PBS -l mppwidth=96` states that the job will be parallelized to run on 96 processors simultaneously.

The comment `#PBS -l mppnppn=32` states that the number of processors per node will be 32. Hence altogether this job will take 3 nodes. Note that 32 is the maximal number of processors that can be run on a node. In practice sometimes it is recommended that the nodes are not fully loaded with processes, but this depends on the sort of jobs and is a delicate matter; here we adopt the simplest solution.

The comment `#PBS -l walltime=00:10:00` does "what it says on the tin": it states that the maximal time that job will be allowed to run is 10 minutes. If the job does not finish by then, it will be killed. It makes sense to put it slightly longer than the expected duration of the run, but not much longer, as this amount affects scheduling: long jobs may have to wait longer before they will be allowed to start.

The comment `#PBS -A e203` is very important because **we will need to amend this**. This defines the name of the project to which the cost of this job will be charged. At the moment of creating the bootable USB sticks this name was not yet know, so in this place some arbitrary name is put, which will need to be changed. On top of that, we will need to direct our script to a special queue, reserved specifically for our workshop (otherwise our job may stuck in general queues for hours). So, we will need to change the line:

```
#PBS -A e203
```

to two lines:

```
#PBS -A d26
#PBS -q R1499583
```

Further changes that need to be done: replace

```
cd /home/e203/e203/ivb203/work/FitzHughNagumo_model
```

with

```
cd /home/d26/d26/<username>/work/parallel_Hector/
```

— this is the directory where you will put all the files and from where you will launch the jobs.

Ignore the line

```
ulimit -s unlimited
```

— i.e. leave it as it is.

Finally,

```
aprun -n 96 -N 32 ./Beatbox fhn_ffr_xz.bbs -profile -verbose -nograph
```

is the line that defines the job. It starts with `aprun` which is the name of MPI wrapper, i.e. a system executable which will start simultaneously the necessary number of copies of BeatBox. Its options are `-n 96` for number of processes to be run simultaneously, and `-N 32` for the number of processes per one node. **These should be the same numbers as in the special comments discussed above**. Finally, goes the usual BeatBox command line. Some notes:

- The name of the executable is `Beatbox` rather than `Beatbox_SEQ` as before: now we are running the parallel version at last!
- The name of the executable is preceded with `./` to make it explicit that the executable will be in the current directory.
- The BeatBox options `-profile` and `-verbose` work the same way as in the sequential version, and are needed to look into details of the work and analyse the performance of BeatBox. In real life production runs, when everything has been perectly tuned, these options may be omitted, for better performance.
- The BeatBox option `-nograph` should switch off any real-time graphics. The real time graphis will not work in parallel runs anyway (the devices ignored), and the BeatBox scripts we are going to run do not have any run-time graphics anyway, but this option does not affect performance so it is good practice to always put it there, "just in case".

Task: Do the above described changes, of the project name and the user name, in both `pbs` files.

Copying files to the remote computer

From now on, it is more convenient to proceed using two terminal window. In the menu of your existing terminal window, click on `File - New Window`. Arrange the two terminal windows so it is easy for you to switch between them. One of them will be logged to the remote computer and will be referred to as *remote terminal*, the other for copying between the local computer and the remote computer, and will be referred to as *local terminal*.

In the terminal window you designate as the **remote terminal**, do the login:

```
[...]$ ssh <username>@login.hector.ac.uk
Password: <password>
Last login: ...
...
<username>@hector-xe6-7:~>
```

At the moment of writing these instructions, we are not aware whether your user directory on HECToR will be cleaned up after the previous user. So these instructions are written without any assumptions about that. One important thing to check is that there is `work` directory there:

```
...>  cd
...>  ls -dl work
lrwxrwxrwx 1 <username> d26 21 Jul  9  2012 work -> /work/d26/d26/<username>
...>
```

and that there is *no* subdirectory `prallel_Hector` in it:

```
...>  cd ~/work
...>  ls -dl parallel*
ls: cannot access parallel*: No such file or directory
...>
```

If you get something like this, it is probably safe to proceed to the next step. If not, some action may be required: call one of the demonstrators for help.

The last thing we do before copying is to create the directory in which we will be working:

```
...>  cd ~/work
...>  mkdir parallel_Hector
...> ls -dl parallel_*
drwxr-sr-x 2 <username> d26 4096 Jun 23 16:41 parallel_Hector
...>
```

In the local terminal, do this:

```
[...]$ cd ~/parallel_Hector
[...]$ scp -pr *.bbg *.rec *.bbs *.pbs \
  <username>@login.hector.ac.uk:work/parallel_Hector/
Password: <password>
ffr.bbg                          100%   12MB 163.5KB/s   01:15
fhn1_NegativeTension.rec         100%   34KB  34.0KB/s   00:00
...
[...]$
```

This will take a while to complete, mainly because of the big file `ffr.bbg`.

Finally, we will need our executable in the working directory. In the **remote window**, do this:

```
...> cd ~/work/parallel_Hector/
...> cp -p /usr/local/packages/budgets/bin/Beatbox ./
...>
```

Now check that everything is in place:

```
...> ls -l
total 20560
-rwxr-xr-x 1 guest21 d26  9359907 Jun 21 15:52 Beatbox
-rwxrwxrwx 1 guest21 d26 12556921 Sep 28  2011 ffr.bbg
-rwxrwxrwx 1 guest21 d26    34808 Apr  4 17:55 fhn1_NegativeTension.rec
-rwxrwxrwx 1 guest21 d26    34106 Apr  4 18:36 fhn1_PositiveTension.rec
-rwxrwxrwx 1 guest21 d26     4711 Apr  5 09:56 fhn3_NegativeTension.bbs
-rwxrwxrwx 1 guest21 d26      375 May 10 12:30 fhn3_NegativeTension.pbs
-rwxrwxrwx 1 guest21 d26     2459 Apr 24 16:01 fhn_ffr_xz.bbs
-rwxrwxrwx 1 guest21 d26      365 May 10 13:43 fhn_ffr_xz.pbs
-rwxrwxrwx 1 guest21 d26     2246 Jun 10 15:06 fhn_ffr_xz_scaling.bbs
...>
```

Submitting the jobs

To submit a BeatBox script, we send the corresponding `pbs` script to the queue:

```
...> qsub fhn_ffr_xz.pbs
...>
```

and similarly for the other `pbs` script. Perhaps it would be wiser to ensure that the first job runs ok before submitting the second (at least in real life).

To monitor the progress of your jobs, do this

```
...> qstat -u $USER
```

from time to time.

In case something goes horribly wrong, it is possible to kill the job, whether queuing or running, by

```
...> qdel [Job-ID]
```

where `[Job-ID]` is the one provided by the `qstat` command.

When the job has finished, you will no longer see it with the `qstat` command. Check the working directory for the outputs expected from the job, and also for the files like `<jobname>.oXXXXXX` and `<jobname>.eXXXXXX`, where `<jobname>` is the name of the job as specified by the `#PBS -N` comment in the `pbs` script, and `XXXXXX` is a unique number provided by the system. The file `<jobname>.oXXXXXX` is the standard output, and `<jobname>.eXXXXXX` is the standard error from the job.

You may inspect the contents of the resulting files on the remote computer, say using `more` command, but for anything more substantial, say visualization, it is better copy it over to your local computer.

Copying the results back to the local computer

.

Before we do that, we note that your working directory on the local computer contains results of our runs of the same jobs. In the **local terminal**, do the following:

```
[...]$ ls -ld *.dir
drwxr-xr-x.  2 beatbox  beatbox  16384 Jun 12 19:37 fhn3_NegativeTension.dir
drwxr-xr-x.  2 beatbox  beatbox   4096 Jun 12 19:37 fhn_ffr_xz.dir
[...]$
```

Since these are exactly the same names as the directories created by HECToR runs, it is a good idea to save these directories for subsequent comparison, by renaming them, say:

```
[...]$ mv fhn3_NegativeTension.dir fhn3_NegativeTension.old
[...]$ mv fhn_ffr_xz.dir fhn_ffr_xz.old
[...]$
```

After that, the easiest way to copy *everything* from the work directory on HECToR to local computer, without copying something that is already here, is to use rsync:

```
[...]$ pwd
/beatbox/parallel_Hector
[...]$ rsync -vazu <username>@login.hector.ac.uk:work/parallel_Hector/ ./
Password: <password>
...
```

After this transfer has, hopefully, successfully completed, you can inspect and visualize the output files. The ppm files in fhn3_NegativeTension.dir and fhn_ffr_xz.dir, as well as in fhn3_NegativeTension.old and fhn_ffr_xz.old are the same sort of files as discussed in the subsection "FitzHugh-Nagumo in a 3D ventricular geometry", and can be visualized with the same methods. You would only need to copy the necessary visualizer scripts to the current directory:

```
[...]$ cp -pr ../FitzHughNagumo_model/*.pl ./
```

and then proceed with view_ffr.pl, surface.pl and inside.pl as before.

Do that and compare the outputs in *.dir and *.old directories.

END