"Introduction to *run_test* script and curvilinear grids"
A tutorial for publication at:
http://polar.ncep.noaa.gov/waves/workshops/WinterSchool2013/

We'll assume that you already have your directories in place, and your main WW3 directory looks like this:

```
$ ls
aux exe guide mod oldtests tmp wwatch3.env bin ftn inp manual obj
regtests work
```

Set up an environment variable that points to this directory. For example, it might be:

```
WW3PATH=/home/user_name/WW3/WW3_Winter_School_Jan2013/ww3/ww3_v4.8/
```

1 Curvilinear grids

It is assumed that, by this point, you have already studied the powerpoint presentation about curvilinear grids, so you don't need a lot of documentation here. Instead, the design of curvilinear grids is demonstrated by example.

If you want details about the implementation in WW3, you can get that here:
http://www7320.nrlssc.navy.mil/pubs/2009/rogers1-2009.pdf
(NRL report by Rogers and Campbell)

Change to the `/regtests/matlab/` directory

```
cd ./regtests/matlab/
```

Start matlab.

Run the example script provided.

```
>> write_x_and_y_example
```

This creates and plots a curvilinear grid. Have a look at the .m script. In this file, you will find a brief discussion of curvilinear grids, and the output files are explained.

Exercise: manipulate the plots; experiment with different configurations by modifying write_x_and_y_example.m

In its initial state, the .m script creates two files, xgrd.2.1c.dat ygrd.2.1c.dat. We'll be using these files (actually, identical files from the repository) in the run_test exercise.

2 The run_test script

Change to the /regtests/ directory

```
cd $WW3PATH/regtests
```

In this exercise, we'll be using the test case called "ww3_tp2.1".

Now, run the "vanilla" version of tp2.1:

```
$ ./bin/run_test $WW3PATH ww3_tp2.1
```

This script (run_test) will prompt you to create the necessary wwatch3.env file if it does not already exist. If this is a fresh directory installed using the provided install script, you will probably get error messages since your “comp” and “link” files are not set yet. If that is the case, no problem, just tell run_test to use a particular compiler. For example, I want to use comp.Portland, so I use:

```
$ ./bin/run_test -c Portland $WW3PATH ww3_tp2.1
```

Now, it should run successfully....but, if you are using Portland, you may notice that it is taking forever to compile w3iogrmd. This is because the Portland compiler is slow when it comes to compiling modules. To fix this, edit the comp.Portland file,

```
emacs $WW3PATH/bin/comp.Portland # replace emacs with gedit or vi, if you
prefer
```

..and swap the comments of these two lines

```
#  opt="-c -Mlist -module $path_m" ; fast="yes"
  opt="-c -Mlist -fast -module $path_m"
```

Try it again.

```
$ ./bin/run_test -c Portland $WW3PATH ww3_tp2.1
```

Now, it should run successfully *and* quickly. Output will go to

```
$WW3PATH/regtests/ww3_tp2.1/work/
```

Notice that we have a number of different switch files to experiment with:

```
$ ls ww3_tp2.1/input/sw*
ww3_tp2.1/input/switch          ww3_tp2.1/input/switch_PR2
ww3_tp2.1/input/switch_PR3_MPI
ww3_tp2.1/input/switch_PR1      ww3_tp2.1/input/switch_PR2_MPI
ww3_tp2.1/input/switch_PR1_MPI  ww3_tp2.1/input/switch_PR3
```

Experiment with different propagation methods:

```
$ ./bin/run_test -s PR1 -w work_PR1 $WW3PATH ww3_tp2.1
$ ./bin/run_test -s PR2 -w work_PR2 $WW3PATH ww3_tp2.1
$ ./bin/run_test -s PR3 -w work_PR3 $WW3PATH ww3_tp2.1
```

Notice that

a) since we gave a -s flag, different switch files are used, switch_PR2 and so forth

b) since we gave a -w flag, output is sent to `$WW3PATH/regtests/ww3_tp2.1/work_PR2/` and so forth.

I'll mention at this point: if you just go through this document copy-and-pasting into your terminal, you can really fly through this tutorial...without learning anything. I recommend to have a look through the files in the work directories, and view the contents of the ascii files, to understand what happened as a result of these run_test commands.

Experiment with mpi next:

```
$ ./bin/run_test -s PR3_MPI -p mpirun -n 3 -w work_PR3_MPI $WW3PATH
ww3_tp2.1
```

Notice that we have a number of different grid files to experiment with.

```
$ ls ww3_tp2.1/input/*grid*inp
```

```
ww3_tp2.1/input/ww3_grid_a.inp    ww3_tp2.1/input/ww3_grid_b_pseudo.inp
ww3_tp2.1/input/ww3_grid_b.inp  ww3_tp2.1/input/ww3_grid_c.inp
```
(also ww3_grid.inp which is identical to ww3_grid_a.inp)

We already ran the standard case, which uses ww3_grid.inp. We can re-run it and send the output to a specific work directory this time:
```
$ ./bin/run_test -g a -w work_grid_a $WW3PATH ww3_tp2.1
```
In my shorthand, this is test case tp2.1a.
And we can run the other grids:
```
$ ./bin/run_test -g b -w work_grid_b $WW3PATH ww3_tp2.1
$ ./bin/run_test -g b_pseudo -w work_grid_b_pseudo $WW3PATH ww3_tp2.1
$ ./bin/run_test -g c -w work_grid_c $WW3PATH ww3_tp2.1
```

In my shorthand, these are test cases tp2.1b, tp2.1bp, and tp2.1c, respectively.
Descriptions of these cases:
tp2.1a : small regular grid (43x43), 24 directions,with deep water in places, land in other places, and the land corresponds to an inlet+jetty configuration
tp2.1b : larger regular grid (273x274), 12 directions, with deep water everywhere.
tp2.1bp : identical to tp2.1b except that the regular grid is specified and read in as if it was a curvilinear grid.
tp2.1c : a true curvilinear grid 226x331 : this is the grid that you created when you ran write_x_and_y_example.m

3 Visualizing output with Matlab

Some matlab scripts are provided with this tutorial:
```
read_scalar_v2.m
read_outf_hs_generic_v2.m
```
I don't know where the short course organizers will place these files, so for sake of this tutorial, let's pretend they are in a directory "~/mfiles/"
You'll need access to these in your work directories. There are a number of ways to do this.
1) copy to the work directory, from work*, `cp ~/mfiles/read_scalar_v2.m . # etc.`
2) link to the files, from work*, `ln -s ~/mfiles/read_scalar_v2.m # etc.`
3) use matlab's "path" command : `path(path,'~/mfiles/')`

3.1 grid a

Recall the case description :
tp2.1a : small regular grid (43x43), 24 directions,with deep water in places, land in other places, and the land corresponds to an inlet+jetty configuration

Now go to the work directory that you just created,
```
$ cd $WW3PATH/regtests/ww3_tp2.1/work_grid_a
```

Notice the output files there:
```
$ ls
```

```
log.ww3      out_grd.ww3  test.ww3         ww3.68060601.hs
ww3.68060603.hs  ww3.68060605.hs  ww3_outf.out            ww3_shel.out
mod_def.ww3  restart.ww3  ww3.68060600.hs  ww3.68060602.hs
ww3.68060604.hs  ww3_grid_a.out   ww3_outf_flds_hrly.out  ww3_strt.out
```

start matlab, and run these commands:

```
time_filename=datenum(1968,06,06,0,0,0);hmax=1.0;dt=1/24;axisin=[];
ext='hs';variablename='SWH';units='m';plot_bathy=0;
read_outf_hs_generic_v2(time_filename,hmax,dt,axisin,ext,variablename,uni
ts,plot_bathy)
```

You will see plots of that inlet+jetty configuration, but the SWH is zero everywhere. This isn't what we wanted.

This happened because our ww3_strt.inp file is not set up for case "a". The initial SWH feature is somewhere off-grid.

Let's do this better.

```
cd $WW3PATH/regtests/ww3_tp2.1/input
emacs ww3_strt.inp
```

and set the uncommented line to :

```
   0.040469  0.0001 225. 200   0.E3 999.E3   0.E3 999.E3  2.501
```

Re-run the case.

```
cd $WW3PATH/regtests
./bin/run_test -g a -s PR3_MPI -p mpirun -n 3 -w work_grid_a $WW3PATH
ww3_tp2.1
```

And make the plots again.

```
cd $WW3PATH/regtests/ww3_tp2.1/work_grid_a
in matlab,
time_filename=datenum(1968,06,06,0,0,0);hmax=1.0;dt=1/24;axisin=[];
ext='hs';variablename='SWH';units='m';plot_bathy=0;
read_outf_hs_generic_v2(time_filename,hmax,dt,axisin,ext,variablename,uni
ts,plot_bathy)
```

This should now look like the original designer (Hendrik) intended it to look.

For small grids like this, you actually don't really need matlab. By this point, you've probably already learned how to look at the ascii output. If you haven't already done so, open up ww3_outf.out in a text editor (or just "more" it):

```
more ww3_outf.out
```

3.2 grid b

Recall the case description:
tp2.1b : larger regular grid (273x274), 12 directions, with deep water everywhere.

```
cd $WW3PATH/regtests/ww3_tp2.1/input
```

emacs ww3_strt.inp

and set the uncommented line to :

```
   0.040469  0.0001 210. 200  2.5E+6 100.0E+3 2.5E+6 100.E+3  2.501
```

Re-run the case.

```
cd $WW3PATH/regtests
./bin/run_test -g b -s PR3_MPI -p mpirun -n 3 -w work_grid_b $WW3PATH
ww3_tp2.1

cd $WW3PATH/regtests/ww3_tp2.1/work_grid_b
in matlab,
time_filename=datenum(1968,06,06,0,0,0);hmax=3.0;dt=1/24;axisin=[2e+6
3.2e+6 2e+6 3.2e+6];
ext='hs';variablename='SWH';units='m';plot_bathy=0;
read_outf_hs_generic_v2(time_filename,hmax,dt,axisin,ext,variablename,uni
ts,plot_bathy)
```

The plots should show the signal moving toward the upper right.
(Direction is set as thm=210.0 in ww3_strt.inp)

3.3 grid c

Recall the case description :
tp2.1c : a true curvilinear grid 226x331: this is the grid that you created when you ran write_x_and_y_example.m

```
cd $WW3PATH/regtests/ww3_tp2.1/input
emacs ww3_strt.inp
```

and set the uncommented line to :

```
   0.040469  0.0001 210. 200  4.5E+6 100.0E+3  4.5E+6 100.E+3  2.501
```

Re-run the case.

```
cd $WW3PATH/regtests
./bin/run_test -g c -s PR3_MPI -p mpirun -n 3 -w work_grid_c $WW3PATH
ww3_tp2.1
```

Now go to the work directory that you just created,

```
cd $WW3PATH/regtests/ww3_tp2.1/work_grid_c
```

Notice the output files there:

```
$ ls
log.ww3      test.ww3         ww3.68060602.hs  ww3_grid_c.out
ww3_shel.out
mod_def.ww3  ww3.68060600.hs  ww3.68060603.hs  ww3_outf.out
ww3_strt.out
out_grd.ww3  restart.ww3      ww3.68060601.hs  ww3.68060604.hs
ww3_outf_flds_hrly.out
```

We'll need matlab scripts to plot the output in the .hs files. Use the files provided with this tutorial again.

Start matlab, and run these commands:

```
time_filename=datenum(1968,06,06,0,0,0);hmax=1.0;dt=1/24;axisin=[];
ext='hs';variablename='SWH';units='m';plot_bathy=0;
read_outf_hs_generic_v2(time_filename,hmax,dt,axisin,ext,variablename,uni
ts,plot_bathy)
```

This will make some strange looking plots, since the matlab script is not very sophisticated: it assumes that the grid is regular. So let's make our own plots from the .mat file that the script just made, plus the grid info that we get from /input/.

(this is a relatively long sequence of commands, so I've also provided them as plot_2p1c.m)

```
clear
colormap(jet)
load SWH.OUTF.mat
xgrd=load('../input/xgrd.2.1c.dat');
ygrd=load('../input/ygrd.2.1c.dat');

figure(1),clf,hold off
for itime=1:6
  subplot(2,3,itime)
  pcolor(xgrd,ygrd,SWH_t{itime}')
  shading interp
  caxis([0 3]);colorbar
  axis square
  title(datestr(time(itime),0))
end

figure(2),clf,hold off % same thing, but zoomed in more
for itime=1:6
  subplot(2,3,itime)
  pcolor(xgrd,ygrd,SWH_t{itime}')
  shading interp
  caxis([0 3]);colorbar
  title(datestr(time(itime),0))
  axis square
  axis([43e+5 49e+5 43e+5 50e+5])
end
```

In the 2nd plot, we can see the signal moving toward the upper right.
(Direction is set as thm=210.0 in ww3_strt.inp)
It doesn't move much, since we only ran this simulation for 5 hours.

Simple exercise: extend simulation to 1 or 2 days, run using run_test w/mpi, and plot.

Credits and contact info:
this document: Erick Rogers (NRL-SSC), `erick.rogers@nrlssc.navy.mil`
run_test script: Tim Campbell (NRL-SSC), `tim.campbell@nrlssc.navy.mil`