Particle filters practical

PA Browne
Email: p.browne@reading.ac.uk
Department of Meteorology, University of Reading,
RG6 6BB, UK

18 September 2014

In this practical we will investigate using different particle filters to assimilate data into the barotropic vorticity model. Recall that the model is solved in 2d on a square with periodic boundary conditions (i.e. a torus). There are 256 grid points in each direction, giving a total of 65,536 state variables.

The model is setup so that only every other grid point in both the x and y directions are observed. Hence there are 16,384 observations each time, and these occur every 50 model timesteps.

These observations were created when you ran the version of the model that we consider the truth earlier in the course.

0 Getting started

Log into windows (somehow).

Once you are finally in - load an xterminal window from the following menus:

Start → All Programs → Cygwin-X → XWin-Server

The may take a while to load. A white window should open up.

Log into ARCHER. Replace `USERNAME` below with the one you were given on a separate piece of paper on Monday.

```
:~> ssh -X USERNAME@login.archer.ac.uk
```

Should you have to log out and log back into ARCHER, and you get an warning/error message that doesn't let you in, try running the following command to help:

```
:~> rm -rf .ssh
```

If you were not here for the truth run, or indeed if for some reason the truth run failed (see below), you may take a copy of the truth run and observations by running the following command:

```
:~> cp -r /work/n02/n02/pbrowne/dacourse /work/n02/n02/$USER
```

Load the necessary python modules.

```
:~> module load numpy matplotlib
```

Move to your /work directory. This is where we shall do everything in this practical.

```
:~> cd /work/n02/n02/$USER/dacourse/bv
```

1 SIR filter

Start python on ARCHER

```
:~> python
```

```
>>> from O_controlsBV import *
```

```
>>> O_setBV(met="SIR")
```

Exit python

```
>>> exit()     OR     Ctrl+d
```

This updates pf_parameters.dat which will control the fortran programs.

Now we must run an ensemble of models for the filter.

Generate a submission script for ARCHER for this by running the command

```
:~> ensemble 48
```

This will generate a submission file `pbs_jobscript` setup with 48 ensemble members.

Submit this to the queue on ARCHER using the command

```
:~> qsub -q R546971 pbs_jobscript
```

You can watch the status of the queue with the command

```
:~> qstat
```

more specifically, just your own jobs in the queue can be shown with the command

```
:~> qstat -u $USER
```

Note the letter in the penultimate column. Q means queuing, R means running, E means ending.

To check the progress of the sequential method, run the following command.

```
:~> wc -l pf_out_00
```

The first number is one larger than the number of timesteps that have been completed. The model is set to run 1200 timesteps, so we have to wait for this number to reach 1201.

When the job has finished, we can now look at the results. We shall do so in an interactive session on one of the main processing nodes on ARCHER.

```
:~> qsub -q R546971 -IVl select=1,walltime=0:30:0 -A n02-training -X
```

```
:~> cd /work/n02/n02/$USER/dacourse/bv
```

```
:~> python
```

```
>>> from 0_controlsBV import *
```

```
>>> O_plotsBV(met="SIR",M=48)
```

There should be a number of outputs created. There are plots (Mean_*.png) showing the truth and the ensemble mean (and their difference). What would you expect the ensemble mean to look like in relation to the truth?

There are plots (Var_timeStep*.png) of the square of the difference between the truth and the mean, in comparison with the variance in the ensemble. What can we deduce from these?

There are plots of the pdfs (Pdfs.png) of 6 different variables at a certain time in the run. Marked on these plots are the truth. How is the data assimilation performing at this point?

There are plots of trajectories (Trajs_*.png) of 9 different variables throughout time. In black the true model state is plotted. If that particular variable is observed, red crosses mark the values of the observations. In blue all the different ensemble members are shown. Can you understand the behaviour in this case?

Finally the rank histogram is plotted (Histogram.png). Remember, if the rank histogram is flat, the truth is indistinguishable from any of the ensemble members. If the rank histogram is *hump shaped* then the ensemble is overdispersive. If the rank histogram is *U shaped* then the ensemble is underdispersive.

When finished looking at the output, to exit press

```
>>> Ctrl+d
```

Should you wish to save these plots to compare later, let us prefix them.

```
:~> for f in *.png; do mv $f sir_$f ; done
```

Now exit the interactive job.

```
:~> Ctrl+d
```

2 Equivalent weights particle filter

Start python on ARCHER

```
:~> python

>>> from O_controlsBV import *

>>> O_setBV(met="ewpf")
```

Exit python

```
>>> exit()     OR     Ctrl+d
```

This updates pf_parameters.dat which will control the fortran programs.

Now we must run an ensemble of models for the filter.

Generate a submission script for ARCHER for this by running the command

```
:~> ensemble 48
```

This will generate a submission file `pbs_jobscript` setup with 48 ensemble members.

Submit this to the queue on ARCHER using the command

```
:~> qsub -q R546971 pbs_jobscript
```

When the job has finished, we can now look at the results. We shall do so in an interactive session on one of the main processing nodes on ARCHER.

```
:~> qsub -q R546971 -IVl select=1,walltime=0:30:0 -A n02-training -X

:~> cd /work/n02/n02/$USER/dacourse/bv

:~> python

>>> from O_controlsBV import *

>>> O_plotsBV(met="ewpf",M=48)
```

When finished looking at the output, to exit press

```
>>> Ctrl+d
```

Now exit the interactive job.

```
:~> Ctrl+d
```

There are a few parameters which you can play with in the equivalent weights particle filter scheme. These are the nudging factor and the proportion of particles kept in the equivalent weights step.

If time allows, experiment with changing these parameters. To do so, we use the O_setBV code in python. By default, these have been set as follows.

```
>>> O_setBV(met="ewpf",keep=0.8,nudgefac=0.05)
```

Change the numbers when you run this line in order to change these parameters.

Also, experiment with changing the number of ensemble members that you run.