Implementing the Correlated Pseudo Marginal in Ox: Documentation

February 22, 2018

Abstract

This document provides a guide to running the examples in the paper "*The Correlated Pseudo-Marginal Method*" . The document together with the code should allow a user who is fairly unfamiliar with the language Ox to quickly perform estimation and analysis. We also provide some tips and explanation for users who are more familiar with either the methodology or the language of Ox. A free (console) version of Ox is available with a few graphical restrictions which are highlighted. The mutivariate state-space examples require the additional installation of the numerical mathematics CGAL C++ library which needs to be compiled. This procedure is briefly explained in this guide and links are provided to more thorough explanation on the CGAL library website.

1 Notes on implementation

The code to implement the examples presented in Section 5 of the main paper has been written in the interpreted econometric matrix language Ox. OxMetrics 4.04 is available at `http://www.doornik.com/download.html`. The free console version of Ox (version 4) is available at the bottom of this webpage. Alternatively the direct link for this version is at `http://www.doornik.com/download/oxmetrics4/b736gw/oxcons.html`. A good introduction to Ox is provided at `http://www.doornik.com/ox/OxIntro.pdf` whilst further documentation is available at `http://www.doornik.com/`. The interpreted language is pretty similar to C and C++ and in fact provides an object oriented programming structure if required. Whilst expertise in Ox would obviously be helpful, the aim of this guide and the associated code is to allow users with no knowledge of the language to be able to run the code, interpret the output and make minor changes. For example, we obviously aim to allow the user to input their own dataset and to specify the length of the time series they wish to estimate.

The random effects model of Section 2 (Section 5.1 of the main paper) requires no further libraries. The univariate state space models of Section 3.1 (Section 5.2 of the main paper) only require efficient routines (from C) for the resampling which are provided by the dynamic library `ASIRSMOOTH.dll`. This should work under Windows and require no further compilation.

The multivariate state space models of Section 2.2 (Section 5.3 of the main paper) require the `CGAL` library to implement the Hilbert sort procedure. The free `CGAL` package and associated libraries (such as `BOOST`) need to be downloaded and compiled on the machine running the code. For Windows, this also involves downloading `cmake` and a free version of Visual Studio. A more detailed description of this process is given in Section 1.2, and a detailed description for installation of the CGAL library is given at `https://doc.cgal.org/latest/Manual/installation.html`.

1.1 A few useful Ox commands

Ox is a matrix orientated language (like Matlab). For detailed information, please consult `http://www.doornik.com/`. We discuss here a few useful commands which may be inserted into the Ox code. These are commands which can be added (as a line) to the code. The free (console) version of Ox is relatively simple to install and run. The index of functions is given in `http://www.doornik.com/ox/`. One of the main differences from the paid version is that there is no graphical front end (for example to show MCMC iterations every few iterations). However there are useful functions for saving graphical and numerical output as detailed below:

- **Printing**: Any output can be printed to the screen. The line command would be:

  ```
  print(mX);
  ```

  This would print the object (say a matrix mX) to the screen.

- **Commenting**: Commenting out commands leads to them being ignored in the program which can be useful. These show up in the colour green in the Ox editor. There are two types of comments. The first is:

  ```
  // print(mX);
  ```

  The second is: `/* print(mX); */`

  The first type is useful for commenting out line by line, whilst the second stops whole chunks of code (between `/*` and `*/`) being read.

- **Exiting**: The command:

  ```
  exit(1);
  ```

  causes the program to stop. Quite useful to print something and then halt the program to check what is going on.

- **Data input**: loadmat("sp500.mat") or loadmat("sp500.xls") is useful for reading in data.

- **Data output:** The function savemat(.) can be used to save a matrix to an Excel file. See `http://www.doornik.com/ox/`.

  For example, the code may be of the form:

  ```
  decl mu = 4.4;
  decl mY = ranpoisson(2, 50, mu); // generate 2 by 50 matrix of Poisson random variables, mean 4.4
  savemat("my_poiss.xls",mY); // saves matrix to excel files (other formats available see index)
  ```

- **Graphical output**: For users with the full version of Ox (Ox Professional) graphical output will be displayed and can easily be saved in a variety of formats (e.g. eps, png, pdf). For users of the free console version a very useful command is to use:

  ```
  SaveDrawWindow("anyname.eps");
  ```

  For example, the code may be of the form:

  ```
  decl mX = rann(1, 100); // generate 1 by 100 vector of normals
  Draw(0, mX, 0, 1); // draw the graph
  ShowDrawWindow(); // displays graph - on screen if full Ox s/w
  SaveDrawWindow("my_out1.eps"); // saves as eps, could use .pdf, .ps
  ```

  More information is available on `http://www.doornik.com/ox/` (under graphics reference).

1.2 CGAL library

Once Ox 4.04 is installed in Windows, in our case in C:\Program Files (x86)\OxMetrics4\ then the instructions for installing CGAL are provided in `https://www.cgal.org/download/windows.html#GeneralPrerequisites`. Please note that CGAL is only necessary for the multivariate state-space models of Section 3.2. We use version 4.7. It is only necessary to have the CGAL libraries fully compiled under the operating system as no further linking is required. We have successfully implemented CGAL under both Windows 8.1 and Windows 10. The compiling of CGAL libraries is rather intricate and time consuming because a number of libraries are prerequisites, the compiler should be matched to the installer files and the path variables in Windows need to be carefully specified. As is clear from the CGAL instructions, it is necessary to install boost, cmake and a compiler.

We downloaded boost (from the CGAL link) with the installer file boost_1_59_0-msvc-12.0-32.exe placing it in C:\dev\boost_1_59_0. We carefully followed the path modification instructions. The name

of this boost file corresponds to the chosen compiler so we chose Visual Studio 2013 (which rather confusingly is VC 12 in the numbering system). This is freely available at `https://my.visualstudio.com/Downloads?q=VisualStudio2013` (with update 5). Downloading and installing cmake from `https://cmake.org/download/` is easy. Provided the paths are set correctly, and the instructions on `https://www.cgal.org/download/windows.html#GeneralPrerequisites` are followed, the compilation under cmake of the CGAL libraries should be successful.

1.3 Archive contents

The zip folder provided contains some code in Ox for implementing the CPM method for the examples described in the main paper. These files are detailed in the next Section. The folders within the zip file are RandomEffects_uniandmulti/, PF_onedimSDE/, PF_onedim/, and PF_multidim/. The first folder RandomEffects_uniandmulti/ is for Section 2. The folder PF_onedimSDE/ corresponds to Section 3.1. The folder PF_onedim/ provides the implementation for Section 3.2 in one dimension. The folder /PF_multidim/, contains the multivariate particle filter code for Section 3.2 and requires the CGAL library of Section 1.2.

2 Random effects model

2.1 Univariate latent variables

The file `RE_onedim.ox` within the zipped folder Online_Oxcode\RandomEffects_uniandmulti\ implements the MH with exact likelihood, PM and CPM schemes for the univariate random effects model as described in Section 5.1 of the main paper. For all schemes, a random walk proposal is θ for used with t-distributed increments.

The main variables set in the program, see the first couple of lines of `main{ }`, are: T (the number of data points), ψ, β and SIM (number of CPM simulations). For the free console version of Ox, the graphical output can be saved as, for example, an .eps file rather than being displayed on screen, see Section 1.1 in this document. Similarly, the output data may be saved in a file (e.g. xls) to be analysed in other software.

For $\psi = 0.7$, $\beta = 0.7$ and $T = 8192$, corresponding to $N = 63$ and $\rho = 0.9945$, this code provides for example the following output:

```
Summary: rho, N, T:
0.99463 63.000 8192.0
Summary: Exact acceptance and ineff, CPM acceptance and inefficiency:
0.68700 6.6985 0.36825 17.758
mean and var of Z (accepted values)
62.831 142.76
mean and var of R=W-Z
-1.2065 2.3776
Theor Lower Bnd on acceptance prob and actual acceptance prob:
0.30278 0.36825
```

The variables R, W, Z are defined in Section 3.1 of the main paper. This output indicates that $\mathrm{IF}_{\mathrm{MH}} = 6.7$ whilst $\mathrm{IF}_{\mathrm{CPM}} = 17.8$ so that the relative inefficiency, $\mathrm{RIF}_{\mathrm{CPM}} = 2.66$. Provided the full version of Ox is used a graphical window will display, which is similar to Figure 1 in the main paper. As the chain has only been run for 32,000 iterations, the accepted chain histogram is not yet a very good approximation to the underlying density. One of the main results of the paper concerning $R = W - Z$, Theorem 3.1, also appears to hold: R is marginally Gaussian and the successive values of R are uncorrelated. The approximating density and histogram are shown for $\mathcal{N}(R \mid -\kappa^2/2; \kappa^2)$ where $\kappa^2 = \widehat{\mathbb{V}}(R) = 2.38$. It would take approximately 100 times as many particles, $N = 6300$, for the standard PM method to work well. This is because the accepted values of W arise approximately from $\mathcal{N}(W \mid \sigma^2/2, \sigma^2)$ where $\sigma^2 = \widehat{\mathbb{V}}(W)$. The actual mean and variance are given by the above output as 62.831, 142.76 suggesting we should take approximately 100 times the number of particles to get a variance between 1 and 2, the optimal value for the standard PM. The acceptance probability is calculated in the above output as 0.368 which is greater than the lower bound of 0.302, which is calculated as the product of the acceptance probability using the exact MH scheme and $\varrho_{\mathrm{U}}(\kappa) = 2\Phi(-\kappa/2)$, see (35) and (37) in the main paper.

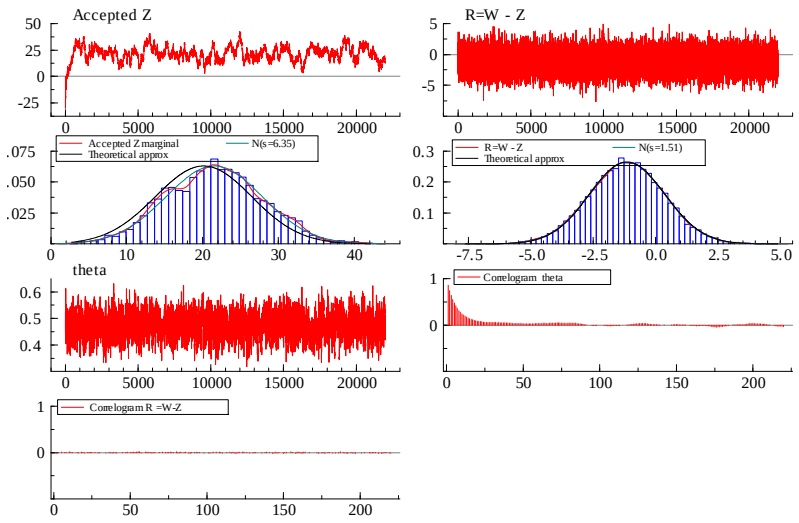

Figure 1: *The CPM output from the Ox file RE_onedim.ox with $T = 1024$ for the univariate random effects model using a random walk proposal.*

Less dramatic improvements can be seen by reducing T by a factor of 8 to 1024, without altering any of the other code so the scaling rule for determining N and ρ remains the same. This also has the advantage of only taking a couple of minutes to run. The graphical output is given below in Figure 1,and here $\sigma^2 = \widehat{\mathbb{V}}(W) = 40.3$ so the CPM only requires about 30 times fewer particles than the standard PM method. However, it can be seen that even with this moderately sized time series, the asymptotic results appear to hold very well.

The value of β can be changed which will alter N and ρ without changing κ much. The resulting computing time RCT=IF$\times N$ can then be calculated. An extremely low value $\beta \ll 0.7$ will result in very slow mixing in the chain despite the acceptance probability being around 0.37. The reasoning for this is provided in Sections 4.3 and 4.4 of the main paper.

The standard PM method can be implemented simply by setting $\rho = 0.0$ in the Ox file. It should be noted that the performance will be very poor unless N is made very large, and proportional to T, the length of the time series. So it is more advisable to do this when T is set to a small number, like 256 in which case $N = 100$ should be sufficient. Note that of course under this setting, whilst R will be marginally Gaussian it will be correlated across draws, as it appears to be in Figure 1 above.

2.2 Multivariate latent variables

The file `RE_multidim.ox` within the zipped folder Online_Oxcode\RandomEffects_uniandmulti\ also implements MH with exact likelihood, PM and CPM schemes for the multivariate random effects model. This model and the results are not reported in the main paper. The simple Gaussian random effects model is very similar to the univariate case. Now we consider a scenario where $d = k = 3$ with $\theta = (\frac{1}{2}, 0, -\frac{1}{2})'$ and

$$X_t \overset{\text{i.i.d.}}{\sim} \mathcal{N}(\theta, I_k), \qquad Y_t | X_t \sim \mathcal{N}(X_t, I_k).$$

The scaling rule is identical to the scalar case, i.e., $N = \beta\sqrt{T}$. For parameter settings $\psi = 0.07$, $\beta = 4$ and $T = 1024$, the output summaries produced by `RE_multidim.ox` are very similar to the univariate case of Section 2.1. The graphical output is given in Figure 2 and the numerical output is summarised in the table below.

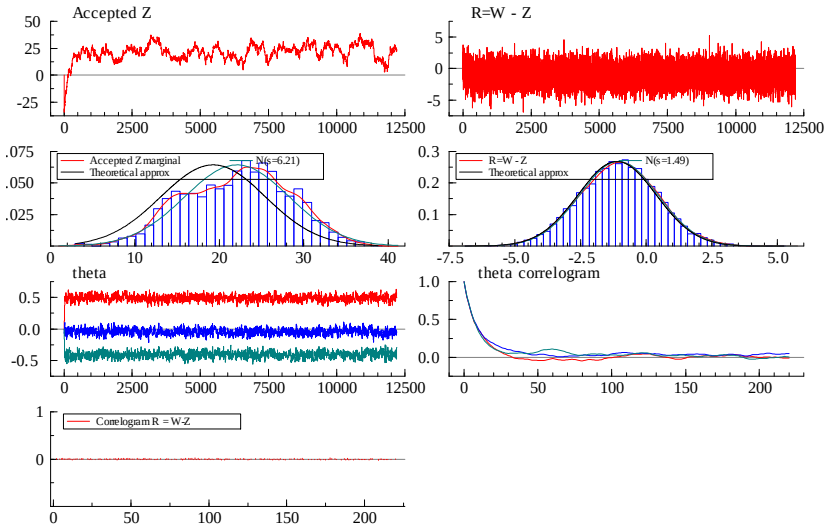

Figure 2: *The CPM output from the Ox file RE_multidim.ox with $T = 1024$ for the 3 dimensional random effects model using a random walk proposal. Bottom right panel shows draws of 3 parameters from the MH with exact likelihood scheme.*

```
Summary: rho, N, T:
0.99129 128.00 1024.0
Summary: Exact acceptance , CPM acceptance:
0.41631 0.27975
Summary: Exact ineff, CPM inefficiency:
10.847 18.485
9.9022 35.758
10.093 31.739
mean and var of Z (accepted values)
22.117 38.581
mean and var of R=W-Z
-1.0471 2.2213
Theor Lower Bnd on acceptance prob and actual acceptance prob:
0.18990 0.27975
```

Increasing T by a factor of 8 to 8192, whilst keeping ψ and β fixed, results in the graphical output of Figure 3. The numerical output summary is also given below.

```
Summary: rho, N, T:
0.99691 362.00 8192.0
Summary: Exact acceptance , CPM acceptance:
0.42052 0.25874
Summary: Exact ineff, CPM inefficiency:
11.297 20.719
10.435 21.484
11.111 23.743
mean and var of Z (accepted values)
64.800 103.76
mean and var of R=W-Z
-1.1695 2.3742
Theor Lower Bnd on acceptance prob and actual acceptance prob:
0.18547 0.25874
```

It is clear that the behaviour of the chain is as expected and the mixing for the 3 parameters is good. The mean of the accepted values of Z indicates that in order to achieve a value of around 1 (near optimal)

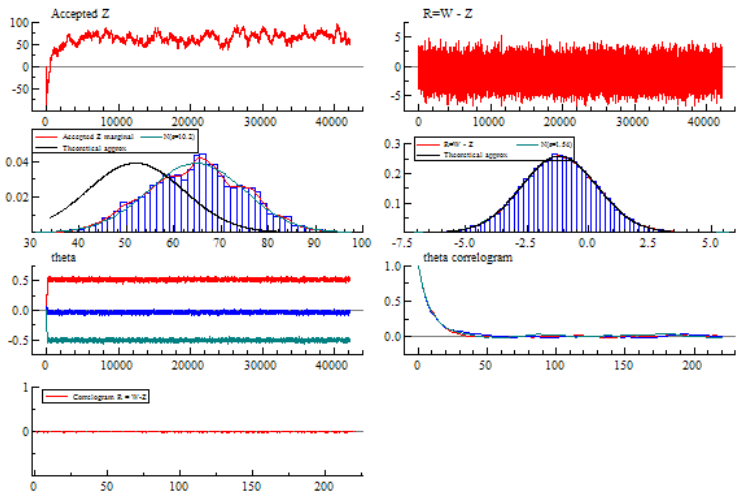

Figure 3: *The CPM output from the Ox file RE_multidim.ox with $T = 8192$ for the 3 dimensional random effects model using a random walk proposal. Bottom RHS plot shows draws of 3 parameters from the MH with exact likelihood scheme.*

for the standard PM we would need to use around 40 times as many particles when $T = 1024$ and 130 times as many particles when $T = 8192$. The inefficiency relative to the MH algorithm using the exact likelihood goes down to only around a value of 2 (for all three parameters) as $T = 8192$.

3 State-space models

3.1 Univariate volatility diffusion

We provide the code, data and basic results for the Heston model of Section 5.2 of the main paper. The Euler discretisation is applied to $x(t) = \log\{\sigma^2(t)\}$ in the (non main) file `ar1noise_class_vol.ox`. We consider the actual data on SP500 daily returns (tablesp500_two.mat) and can estimate both the standard model with no leverage ($\chi = 0$) and the four parameter leverage model. The former requires fewer particles and is therefore quicker to run. We consider various lengths of the time series, T, going back in time from July 7, 2006. The output which is recorded (in a matrix output to a file) is the sequences across iterations of log-likelihood difference , the accepted simulated log-likelihood, the acceptance rate, and the draws of the four parameters $\theta = (\mu, \phi, \omega, \chi)$, where $\phi = e^{-v}$. As described in the paper, a random walk proposal is applied having transformed the parameters to the real line. The main files are described in the following table:

	No leverage	Leverage
Main file	`pmcmc_SQRT_nolev.ox`	`pmcmc_SQRT_lev.ox`
Input data	`tablesp500_two.mat`	`tablesp500_two.mat`
Data output	`output_nolev.mat`	`output_lev.mat`
Graphical output	screen/`out.eps`	screen/`out.eps`

The output file (currently called "output_lev.mat" if we consider the model with leverage) contains the 7 by SIM (number of simulations) matrix. The code for analysis in Ox is `analyse_SDE.ox`, though of course the analysis can be conducted by importing the data in any package.

The CPM is initialised by running 300 independent particle filters (RUN_g variable=300) at a central parameter value obtained from a preliminary run. This allows to estimate σ^2, the variance of the log-likelihood error. This roughly translates as the factor we would need to increase N by in order to effectively run the standard PM approach. After this and until the burn in of 3000 (variable BURN_IN) the parameters are not moved but the U variables are moved within the CPM scheme.

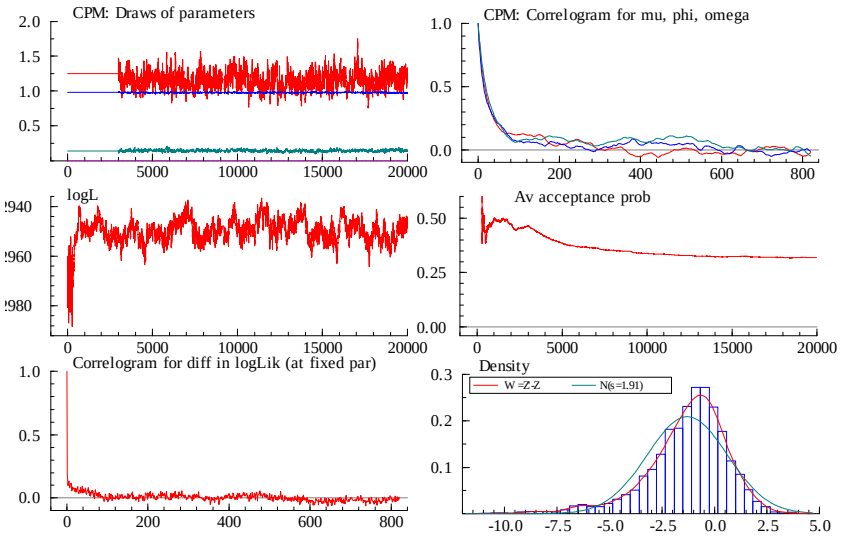

Figure 4: *The results of running the no leverage SQRT volatility model based on the S&P 500 data,* $T = 2000$.

The results from running `pmcmc_SQRT_nolev.ox` with $T = 2000$ with the current settings for β and ψ are displayed in Figure 4. For this moderate length time series with no leverage the gain over the standard PM is over 30 fold. The program takes around 10 minutes to run.

3.2 Linear Gaussian state space models

The code and example in this section corresponds to the examples of Section 5.3 (Linear Gaussian state-space models) of the main paper. The implementation for one dimension is found in the folder \PF_onedim\. in the program file `PF_SSFmult_dim1.ox`. This does not require the CGAL library to be compiled. The multivariate versions do require the CGAL libraries as they employ sorting according to the Hilbert curve. The description below will simply focus on `PF_SSFmult_dim2.ox` in the folder \PF_multidim\.

3.2.1 Basic run

As summarised in the table below, the main file is called `PF_SSFmult_dim2.ox`. Running this file produces output in two data files (see table below). An analysis file `analyse_d2.ox` can then be run which produces graphical output. This output is as described in Section 5.3 and Figure 8 of the main paper. We are currently only taking $T = 400$ so that the analysis can be conducted fairly quickly. A slightly more detailed description is given below.

3.2.2 More details

The main file `PF_SSFmult_dim2.ox` calls other Ox files and contains the definition of many functions. We note that the Hilbert sorting is here applied both before and after the innovation step (twice each time step) within the implementation of the particle filter. This is theoretically valid and we note it slightly improves performance. This may be seen by inspecting the function `PF_loop_U(...)` as defined in the main file.

The functions of the main file `exact_MCMC(...)`, `particle_MCMC(...)` perform exact MH and CPM respectively. In the main() part of the file, the settings are: theta = 0.4, dim = 2, T =400, SIM = 20000, where dim is dimension of the state and SIM is the number of simulations. Experiments indicate that for dim=$1, 3, 4$ the settings (these were found when $T = 6400$ as in the main paper) should be around $(\beta, \psi) = (0.47, 0.5)$, $(1.57, 0.042)$, $(2.61, 0.014)$. As the dimension changes, the value of α is automatically set to $k/(k+1)$.

For the running of the MCMC routines, the variables `SIM` and `SIM0` indicate the number of iterations of the exact MH and the CPM schemes respectively. We initialize the CPM by running `SIM_ini` independent particle filters at a central parameter value. From iterates `SIM_ini` to `SIM_fix`, the scheme performs CPM but with θ fixed. After `SIM_fix`, the CPM scheme works fully making proposals for (u, θ).

Main file	`PF_SSFmult_dim2.ox`
Data output 1	`mPar_MH.mat`, for MH
Data output 2	`mPar_CPM.mat`, for CPM
File for analysis	`analyse_d2.ox`
Data input 1	`mPar_MH.mat`, for MH
Data input 2	`mPar_CPM.mat`, for CPM