USING COMPILED CODE IN `POMP`

AARON A. KING

1. A two-dimensional Ornstein-Uhlenbeck process.

To keep things simple, we will study a discrete-time process. The tricks below will continue to be useful even in the case of a continuous-time process, but the computational effort will be greater. The unobserved Ornstein-Uhlenbeck (OU) process $X_t \in \mathbb{R}^2$ satisfies

$$X_t = A\,X_{t-1} + \xi_t.$$

The observation process is

$$Y_t = B\,X_t + \varepsilon_t.$$

In these equations, A and and B are $2{\times}2$ constant matrices; ξ_t and ε_t are mutually-independent families of i.i.d. bivariate normal random variables. We let $\sigma\sigma^T$ be the variance-covariance matrix of ξ_t, where σ is lower-triangular; likewise, we let $\tau\tau^T$ be that of ε_t.

Since many of the methods we will use require us to simulate the process and/or measurement models many times, it is a good idea to use native (compiled) codes for the computational heavy lifting. The package includes some C codes that were written to implement the OU example. Read the source (file 'ou2.c') for details.

1.1. The `.C` interface.

```
> ou2.rprocess <- function(xstart, times, params,
+     ...) {
+     nvar <- nrow(xstart)
+     npar <- nrow(params)
+     nrep <- ncol(xstart)
+     ntimes <- length(times)
+     parindex <- match(c("alpha.1", "alpha.2",
+         "alpha.3", "alpha.4", "sigma.1", "sigma.2",
+         "sigma.3"), rownames(params)) - 1
+     array(.C("ou2_adv", X = double(nvar * nrep *
+         ntimes), xstart = as.double(xstart), par = as.double(params),
+         times = as.double(times), n = as.integer(c(nvar,
+             npar, nrep, ntimes)), parindex = as.integer(parindex),
+         DUP = FALSE, NAOK = TRUE, PACKAGE = "pomp")$X,
+         dim = c(nvar, nrep, ntimes), dimnames = list(rownames(xstart),
+             NULL, NULL))
+ }
> ou2.dprocess <- function(x, times, params, log,
+     ...) {
+     nvar <- nrow(x)
+     npar <- nrow(params)
+     nrep <- ncol(x)
+     ntimes <- length(times)
+     parindex <- match(c("alpha.1", "alpha.2",
+         "alpha.3", "alpha.4", "sigma.1", "sigma.2",
+         "sigma.3"), rownames(params)) - 1
```

Date: May 14, 2008.

```
+     array(.C("ou2_pdf", d = double(nrep * (ntimes -
+         1)), X = as.double(x), par = as.double(params),
+         times = as.double(times), n = as.integer(c(nvar,
+             npar, nrep, ntimes)), parindex = as.integer(parindex),
+         give_log = as.integer(log), DUP = FALSE,
+         NAOK = TRUE, PACKAGE = "pomp")$d, dim = c(nrep,
+         ntimes - 1))
+ }
> bvnorm.dmeasure <- function(y, x, times, params,
+     log = TRUE, ...) {
+     measindex <- match(c("tau"), rownames(params)) -
+         1
+     nvar <- dim(x)[1]
+     nrep <- dim(x)[2]
+     ntimes <- dim(x)[3]
+     npar <- nrow(params)
+     nobs <- 2
+     array(.C("normal_dmeasure", n = as.integer(c(nvar,
+         npar, nrep, ntimes, nobs)), X = as.double(x),
+         par = as.double(params), index = as.integer(measindex),
+         Y = as.double(y), f = double(nrep * ntimes),
+         give_log = as.integer(log), DUP = FALSE,
+         NAOK = TRUE, PACKAGE = "pomp")$f, dim = c(nrep,
+         ntimes))
+ }
> bvnorm.rmeasure <- function(x, times, params,
+     ...) {
+     nvar <- dim(x)[1]
+     nrep <- dim(x)[2]
+     ntimes <- dim(x)[3]
+     npar <- dim(params)[1]
+     nobs <- 2
+     measindex <- match(c("tau"), rownames(params)) -
+         1
+     array(.C("normal_rmeasure", n = as.integer(c(nvar,
+         npar, nrep, ntimes, nobs)), X = as.double(x),
+         par = as.double(params), index = as.integer(measindex),
+         obs = double(nobs * nrep * ntimes), DUP = FALSE,
+         NAOK = TRUE, PACKAGE = "pomp")$obs, dim = c(nobs,
+         nrep, ntimes), dimnames = list(c("y1",
+         "y2"), NULL, NULL))
+ }
> ou2 <- pomp(times = seq(1, 100), data = rbind(y1 = rep(0,
+     100), y2 = rep(0, 100)), t0 = 0, rprocess = ou2.rprocess,
+     dprocess = ou2.dprocess, rmeasure = bvnorm.rmeasure,
+     dmeasure = bvnorm.dmeasure)
```

We'll specify some parameters:

```
> p <- c(alpha.1 = 0.9, alpha.2 = 0, alpha.3 = 0,
+     alpha.4 = 0.99, sigma.1 = 1, sigma.2 = 0,
+     sigma.3 = 2, tau = 1, x1.0 = 50, x2.0 = -50)
```

```
> tic <- Sys.time()
> ou2 <- simulate(ou2, params = p, nsim = 1000,
```

```
+     seed = 800733088)[[1]]
> toc <- Sys.time()
> print(toc - tic)
```

```
Time difference of 2.712073 secs
```

1.2. **The `.Call` interface.** The following wrapper functions make use of the above compiled codes.

```
> ou2.rprocess <- function(xstart, times, params,
+     ...) .Call("ou2_simulator", xstart, times,
+     params)
> ou2.dprocess <- function(x, times, params, log = FALSE,
+     ...) .Call("ou2_density", x, as.numeric(times),
+     params, log)
> bvnorm.dmeasure <- function(y, x, times, params,
+     log = FALSE, ...) .Call("bivariate_normal_dmeasure",
+     y, x, as.numeric(times), params, log)
> bvnorm.rmeasure <- function(x, times, params,
+     ...) .Call("bivariate_normal_rmeasure", x,
+     as.numeric(times), params)
```

To take advantage of the compiled functions, we need to reconstruct the `pomp` object.

```
> ou2 <- pomp(times = seq(1, 100), data = rbind(y1 = rep(0,
+     100), y2 = rep(0, 100)), t0 = 0, rprocess = ou2.rprocess,
+     dprocess = ou2.dprocess, rmeasure = bvnorm.rmeasure,
+     dmeasure = bvnorm.dmeasure, ivpnames = c("x1.0",
+         "x2.0"), parnames = c("alpha.1", "alpha.2",
+         "alpha.3", "alpha.4", "sigma.1", "sigma.2",
+         "sigma.3", "tau"))
```

Notice that we have added two objects, `ivpnames` and `parnames` to the `pomp` object. These character vectors are placed into the `userdata` slot of the `pomp` object and will be passed to each of the process and measurement model functions. They will come in handy later when we do particle filtering.

We'll fill the data slot with simulated data:

```
> tic <- Sys.time()
> ou2 <- simulate(ou2, params = p, nsim = 1000,
+     seed = 800733088)[[1]]
> toc <- Sys.time()
> print(toc - tic)
```

```
Time difference of 2.609013 secs
```

Fig. 1 plots the data.

Let's make sure everything works.

```
> x0 <- init.state(ou2, params = p)
> x <- rprocess(ou2, xstart = as.matrix(x0), times = c(0,
+     time(ou2)), params = as.matrix(p))
> y <- rmeasure(ou2, x = x[, , -1, drop = F], times = time(ou2),
+     params = as.matrix(p))
> dprocess(ou2, x[, , 36:41, drop = F], times = time(ou2)[35:40],
+     params = as.matrix(p), log = T)
```

```
          [,1]     [,2]      [,3]      [,4]      [,5]
[1,] -1.988691 -4.22445 -3.065896 -1.945323 -2.192079
```

```
> dmeasure(ou2, y = y[, 1, 1:4], x = x[, , 2:5,
+     drop = F], times = time(ou2)[1:4], params = as.matrix(p),
+     log = T)
```

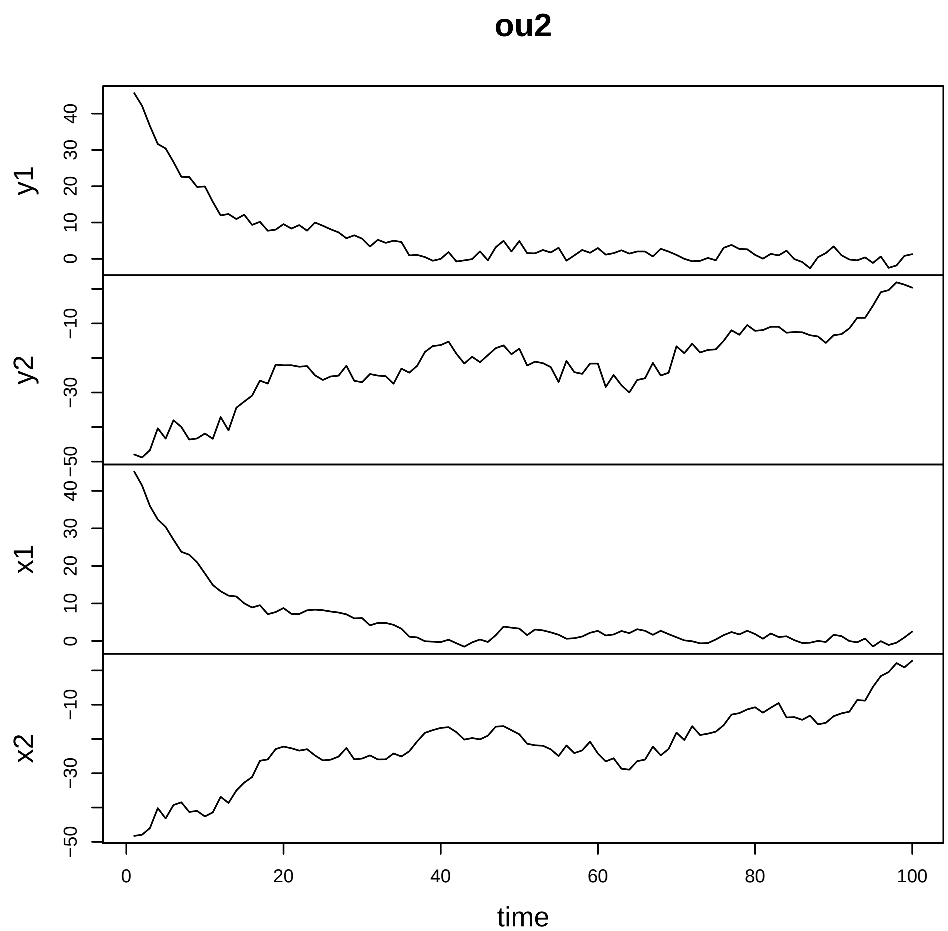

FIGURE 1. One realization of the two-dimensional OU process.

```
          [,1]      [,2]      [,3]      [,4]
[1,] -3.846353 -3.579476 -2.230835 -2.379882
```

The `pomp` object we just created is included in the package: use `data(ou2)` to retrieve it.

A. A. KING, DEPARTMENTS OF ECOLOGY & EVOLUTIONARY BIOLOGY AND MATHEMATICS, UNIVERSITY OF MICHIGAN, ANN ARBOR, MICHIGAN 48109-1048 USA

E-mail address: `kingaa at umich dot edu`

URL: `http://www.umich.edu/~kingaa`