Statistics in Biosciences manuscript No.
(will be inserted by the editor)

A Particle Filter Approach to Multiprocess Dynamic Models with Application to Hormone Data

Supplement B: Example Matlab Code

Ziyue Liu

Department of Biostatistics, Fairbanks School of Public Health and School of Medicine, Indiana University, 410 W. 10TH St., Suite 3000, Indianapolis, IN 46202, USA.

Tel.: +001-317-2786734

Fax: +001-317-2742678

E-mail: ziliu@iu.edu

Table of Contents

Read in data, set path

```
clear
clc
addpath('C:\Users\ziliu.ADS\Documents\mprog\sbpp-gamma')
% read in data
%dat=load('C:\Users\ziliu.ADS\Documents\paper\dmem\data\cort.mat')

%a simulated series as in Guo et al 1999
dat=[34.4869; 31.5246; 30.1842; 27.2080; 27.3270; 25.3538; 25.6737;
   26.8707; 40.1862; 36.3959; 32.8935; 32.9351; 31.4124; 29.7213;
   32.0355; 30.5897; 31.2845; 32.0449; 32.0136; 32.0786; 32.0305;
   43.0864; 40.9459; 36.9226; 49.9756; 45.1780; 51.6372; 49.5080;
   43.5909; 55.6887; 48.7038; 44.1438; 49.4833; 45.5847; 55.0681;
   49.8833; 43.7684; 42.4096; 40.5782; 37.5312; 36.4526; 35.0706;
   36.2236; 48.4704; 43.7940; 39.1879; 36.9465; 36.2339; 36.0744;
   34.1230; 40.4906; 37.1775; 34.6944; 32.2586; 31.3812; 32.7248;
   29.3397; 28.7604; 27.1978; 28.7927; 27.1132; 28.5214; 32.0620;
   29.1151; 26.8689; 27.6379; 35.1122; 31.3488; 28.4624; 26.6647;
   23.4898; 21.9376; 20.3986; 20.4595; 18.6581; 17.3227; 15.1767;
   16.9705; 16.4438; 16.1967; 15.5911; 14.2398; 12.4941; 13.6040;
   12.5489; 12.9998; 11.2400; 11.7456;  9.9510;  9.1123;  8.8719;
   17.9048; 13.6076; 10.5590;  8.7694;  9.2277;  9.2072;  6.8445;
    6.4539;  6.1144; 16.5384; 12.5556; 11.6224;  8.9477;  9.8249;
   15.2266; 12.5618; 10.2398; 19.0942; 11.5815; 12.3536;  7.2226;
    8.1654;  7.6028;  6.7120;  6.6505;  8.4944;  7.2750;  7.1487;
    7.9975;  7.0118;  7.2709;  7.7351;  6.8609;  8.1859;  8.5136;
   10.2088; 11.0113; 10.3338; 10.1901; 11.3667; 13.9568; 23.8260;
   21.3579; 18.7671; 18.7556; 27.1478; 24.5506; 22.5556; 22.3145;
   20.6102; 20.4378; 19.5546; 20.6471];
```

Get initial parameter values from GWB

```
addpath('C:\Users\ziliu.ADS\Documents\mprog\sbpp')
%from Guo etal 1999 JASA paper
theta_ini=zeros(6,1);
theta_ini(1)=7;
theta_ini(2)=log(0.8/0.2);
theta_ini(3)=log(0.1/0.9);
```

```
theta_ini(4)=log(5.12);
theta_ini(5)=log(1.67);
theta_ini(6)=log(0.85);
[xsG,vsG,ppostG,theta_new]=sbpp(dat,theta_ini);

% transform the parameters back
theta=exp(theta_new);
lambda=theta(1);
phi=theta(2)/(1+theta(2));
pprior=theta(3)/(1+theta(3));
mu=theta(4);
sigp=theta(5);
sigerror=theta(6);

gama=mu^2/sigp;               %gamma shape parameter, for pulse
gamb=sigp/mu;               %gamma scale parameter, for pusle
%
%transform to working scale
lpar0=zeros(6,1);
lpar0(1)=log(lambda);
lpar0(2)=log(pprior/(1-pprior));
lpar0(3)=log(phi/(1-phi));
lpar0(4)=log(gama);
lpar0(5)=log(gamb);
lpar0(6)=log(sigerror);
```

Parameter estimation

```
%numer of iterations
num_ite=1000*numel(lpar0);
options=optimset('MaxFunEvals',num_ite,'MaxIter',...
    num_ite,'TolFun',1e-4,'TolX',1e-4);

%number of particles
npart=1000;
%random seed
rseed=55555;
%call optimization routine
[par_new,m2llh]=fminsearch(@(lpar) ...
    llh_sbpp_gamma(lpar,dat,npart,rseed),lpar0,options);
```

Posteriors: particle filtering and smoothing

```
%number of particles
npart2=10000;
%number of repeat in smoothing
nrepeat=1000;
%random seed
rseed2=88888;
%get posterior
[fsmooth,xsmooth,insmooth,ovsmooth,fmfil,fvfil,xmfil,xvfil,mfil,vfil,...
    pfil,inputfil]=post_sbpp_gamma(par_new,dat,npart2,rseed2,nrepeat);
```

Get the fitted components

```
npt=numel(dat);
%overall posterior mean
mu=reshape(mean(ovsmooth,1),npt,1);

%smooth baseline
f1smooth=zeros(npt,nrepeat);
for kkk=1:nrepeat
    f1smooth(:,kkk)=reshape(fsmooth(1,kkk,:),npt,1);
end
baseL=mean(f1smooth,2);

%pulse estiamte
PulseL=reshape(mean(insmooth,1),npt,1);

%pulse probability
ppostL=reshape(mean((insmooth>0)),npt,1);
```

Function to calculate the likelihood

```
function m2llh=llh_sbpp_gamma(lpar,dat,npart,rseed)

%transform parameters to original scale
para=exp(lpar);
lambda=para(1);              %smooting parameter
pprior=para(2)/(1+para(2));  %prior prob of being a pulse
phi=para(3)/(1+para(3));     %AR(1) coefficient
gama=para(4);                %gamma shape parameter, for pulse
gamb=para(5);                %gamma scale parameter, for pusle
sigerror=para(6);            %observational error variance


%number of time point
npt=numel(dat);
%lower limit, log-like, that will be set as zero
lowerb=log(1/npart^10);

%system matrices
dt=1/npt;
H=[1 dt; 0 1];
sigf=lambda*[1/3*dt^3 1/2*dt^2; 1/2*dt^2 dt];


% initialize random number generator
s=RandStream('mt19937ar','Seed',rseed);
RandStream.setGlobalStream(s);
%initial for pulse
xvec=pprior*gama*gamb/(1-phi)+...
    binornd(1,pprior,npart,1).*gamrnd(gama,gamb,npart,1);
%initial for smooth baseline
p0=1000;         %numeric diffuse
```

```
fmat=zeros(2,npart);
for iii=1:1:npart
    fmat(:,iii)=blkdiag(p0^0.5*randn(2,1));
end

% filtering loop
llh=0;
for iii=1:1:npt
    %generate new states
    s=RandStream('mt19937ar','Seed',55555+iii);
    RandStream.setGlobalStream(s);
    %pulse part
    gamnew=(rand(npart,1)<pprior).*gamrnd(gama,gamb,npart,1); %innovation
    xvec=phi*xvec+gamnew;  %decay + innovation
    %spline part
    for jjj=1:1:npart
        fmat(:,jjj)=H*fmat(:,jjj)+(mvnrnd([0 0],sigf))';
    end

    %mean vector, combine baseline and pules
    fvec=reshape(fmat(1,:),npart,1);
    alphavec=xvec+fvec;

    %calculate weights
    %log scale
    lwvec=-0.5*log(2*pi)-0.5*log(sigerror)...
        -0.5*(dat(iii)-alphavec).^2/sigerror;
    %weight, reset so that no NaN, no Inf
    wvec=exp(lwvec-max(lwvec)+lowerb);

    %loglikelihood
    llh_here=log(mean(wvec))+max(lwvec)-lowerb;
    llh=llh+llh_here;

    %rescale weights
    wvec=wvec/(sum(wvec));

    %resample
    indvec=randsample(npart,npart,true,wvec);
    xvec=xvec(indvec);
    fmat=fmat(:,indvec);

end

m2llh=-2*llh;
```

Function to calcualte the posterior items

```
function [fsmooth,xsmooth,insmooth,ovsmooth,fmfil,fvfil,xmfil,xvfil,...
    mfil,vfil,pfil,inputfil]=post_sbpp_gamma(lpar,dat,npart,rseed,nrepeat)

% transform parameters to original scale
para=exp(lpar);
```

```
lambda=para(1);              %smooting parameter
pprior=para(2)/(1+para(2));  %prior prob of being a pulse
phi=para(3)/(1+para(3));     %AR(1) coefficient
gama=para(4);                %gamma shape parameter, for pulse
gamb=para(5);                %gamma scale parameter, for pusle
sigerror=para(6);            %observational error variance


% system items
%number of time point
npt=numel(dat);
%lower limit, log-like, that will be set as zero
lowerb=log(1/npart^10);

%system matrices
dt=1/npt;
H=[1 dt; 0 1];
sigf=lambda*[1/3*dt^3 1/2*dt^2; 1/2*dt^2 dt];

% initialize random number generator
s=RandStream('mt19937ar','Seed',rseed);
RandStream.setGlobalStream(s);
%initial for pulse
xvec=pprior*gama*gamb/(1-phi)...
    +binornd(1,pprior,npart,1).*gamrnd(gama,gamb,npart,1);
%initial for smooth baseline
p0=1000;         %numeric diffuse
fmat=zeros(2,npart);
for iii=1:1:npart
    fmat(:,iii)=blkdiag(p0^0.5*randn(2,1));
end

% matrices to hold
%the filtered values
fmfil=zeros(npt,1);  %smoothing baseline, mean
fvfil=zeros(npt,1);  %                   , variance
xmfil=zeros(npt,1);  %pusatile, cumulative, mean
xvfil=zeros(npt,1);  %     , variance
mfil=zeros(npt,1);   %overall, mean
vfil=zeros(npt,1);   %       , variance
pfil=zeros(npt,1);   %filtered prob of being a pulse
inputfil=zeros(npt,1); %pulse input

%items needed for smoothing
ffil=zeros(2,npart,npt);   %baseline component
xfil=zeros(npart,npt);   %AR(1) component
invec=zeros(npart,npt);  %pulse input
ovfil=zeros(npart,npt);  %overall level

% filtering loop
for iii=1:1:npt
    %generate new states
    %s=RandStream('mt19937ar','Seed',rseed+iii);
    RandStream.setGlobalStream(s);
```

```
    %pulse part
    gamnew=(rand(npart,1)<pprior).*gamrnd(gama,gamb,npart,1); %innovation
    xvec=phi*xvec+gamnew;  %decay + innovation
    %spline part
    for jjj=1:1:npart
        fmat(:,jjj)=H*fmat(:,jjj)+(mvnrnd([0 0],sigf))';
    end

    %mean vector, combine baseline and pules
    fvec=reshape(fmat(1,:),npart,1);
    alphavec=xvec+fvec;

    %calculate weights
    %log scale
    lwvec=-0.5*log(2*pi)-0.5*log(sigerror)...
        -0.5*(dat(iii)-alphavec).^2/sigerror;
    %weight, reset so that no NaN, no Inf
    wvec=exp(lwvec-max(lwvec)+lowerb);
    wvec=wvec/(sum(wvec));

    %mean and variance, filtered
    fmfil(iii)=sum(wvec.*fvec);
    xmfil(iii)=sum(wvec.*xvec);
    mfil(iii) =sum(wvec.*alphavec);
    fvfil(iii)=sum(wvec.*fvec.*fvec)-(fmfil(iii))^2;
    xvfil(iii)=sum(wvec.*xvec.*xvec)-(xmfil(iii))^2;
    vfil(iii) =sum(wvec.*alphavec.*alphavec)-(mfil(iii))^2;
    inputfil(iii)=sum(wvec.*gamnew);

    %resample
    indvec=randsample(npart,npart,true,wvec);
    xvec=xvec(indvec);
    fmat=fmat(:,indvec);

    %filtered prob being a pulse
    pfil(iii)=mean(gamnew(indvec)>0);

    %store items
    ffil(:,:,iii)=fmat;
    xfil(:,iii)=xvec;
    invec(:,iii)=gamnew(indvec);
    ovfil(:,iii)=alphavec(indvec);  %overall level
end



% smoothing loop, Godsill, Doucet and West 2004 JASA paper
% posterior items
fsmooth=zeros(2,nrepeat,npt);   %baseline component
xsmooth=zeros(nrepeat,npt);   %AR(1) component
insmooth=zeros(nrepeat,npt);  %pulse input
ovsmooth=zeros(nrepeat,npt);  %overall level

%spline innovation variance parts
```

```
ldenconst=-log(2*pi)-0.5*log(det(sigf));
invsigf=inv(sigf);
%gamma constant part
gamconst=-gama*log(gamb)-gammaln(gama);

for kkk=1:nrepeat
    %pick one at the last time point
    indn=ceil(npart*rand);
    fhere=ffil(:,indn,npt);
    xhere=xfil(indn,npt);
    inhere=invec(indn,npt);
    ovhere=ovfil(indn,npt);

    %record
    fsmooth(:,kkk,npt)=fhere;
    xsmooth(kkk,npt)=xhere;
    insmooth(kkk,npt)=inhere;
    ovsmooth(kkk,npt)=ovhere;

    %loop through remaining time points
    for iii=npt-1:-1:1
        %log-density of the AR(1) part, contional on t, a mixture
        denar=zeros(npart,1)-Inf;
        netdiff=xhere-phi*xfil(:,iii);

        for jjj=1:1:npart
            if netdiff(jjj)==0
                denar(jjj)=log(1-pprior);
            elseif netdiff(jjj)>0
                denar(jjj)=log(pprior)*(gamconst+...
                    (gama-1)*log(netdiff(jjj))-netdiff(jjj)/gamb);
            end
        end

        %log-density of the splien part, contional on t, a normal
        denss=zeros(npart,1);
        for jjj=1:1:npart
            denss(jjj)=ldenconst-0.5*(fhere-...
                H*ffil(:,jjj,iii))'*invsigf*(fhere-H*ffil(:,jjj,iii));
        end
        %overall log-density
        ldenov=denar+denss;
        %remove ones too small
        denov=exp(ldenov-max(ldenov)+lowerb);
        wihere=denov/(sum(denov));

        %draw one at this time point
        indn=randsample(npart,1,true,wihere);  %index here
        fhere=ffil(:,indn,iii);
        xhere=xfil(indn,iii);
        inhere=invec(indn,iii);
        ovhere=ovfil(indn,iii);

        %record
```

```
            fsmooth(:,kkk,iii)=fhere;
            xsmooth(kkk,iii)=xhere;
            insmooth(kkk,iii)=inhere;
            ovsmooth(kkk,iii)=ovhere;
        end
        %kkk
end
```

Published with MATLAB® R2013a