MATLAB code for EM algorithm

This source code does the foliowing:

- Simulate a Gaussian random field $\mathcal{X}$ with O-U covariance function.
- Simulate randomly missing observations given a fixed level of missingness.
- Simulate a conditional GRF with 'complete' data, where missing observations are replaced with conditional mean based on available observations and fixed parameter value.
- Evaluate the likelihood function of $\mathcal{X}$.
- Implement the EM algorithm to estimate the true parameter values given $\mathcal{X}$ (with missing observations) and initial parameter values.
- An illustrative example that uses the above function to test the EM algorithm.

grf_OU.m : function to simulate a realization of $\mathcal{X}$

```
% Function to simulate Gaussian random field with Ornstein-Uhlenbeck covariance function
% function [X,A,B,Gamma]=grf_OU(u,v,sigma2,mu,lambda)
% INPUT:
% u = horizontal coordinate of input grid
% v = vertical coordinate of input grid
% cov(X(u,v),X(u',v')) = sigma2*exp(-lambda|u-u'|-mu|v-v'|)
%
% OUTPUT:
% X = an N-by-M array of a random field realization (its graph is represented as surface above the meshgrid
% deinfed by the input grid u x v)
% A = Covariance matrix for the 'horizonal' part of the random field
% B = Covariance matrix for the 'vertical' part of the random field
% Gamma = Covariance matrix for the whole random field with complete
% observations
function [X,A,B,Gamma]=grf_OU(u,v,sigma2,mu,lambda)
% Define the absolute distance between inputs from u and inputs from
% v:
M=length(u);
N=length(v);
% Distance between consecutive input points:
eta=zeros(M,1);
nu=zeros(N,1);
```

```
eta(1)=u(1);
nu(1) =v(1);
eta(2:M)=abs(u(2:M)-u(1:M-1));
nu(2:N)=abs(v(2:N)-v(1:N-1));
% Define the component of the covariance based on distance between grid
% points:
a(1)=0;
b(1)=0;
a(2:M)=exp(-lambda.*eta(2:M));
b(2:N)=exp(-mu.*nu(2:N));

% Define the covariance function for horizonal and vertical components:
A = eye(M,M);
B = eye(N,N);

for i=1:M-1
    A(i,i+1)=a(i+1);
 for j=i+1:M

     A(i,j)=prod(a(i+1:j));
     A(j,i)=A(i,j);
 end
end

for i=1:N-1
```

```
    B(i,i+1)=b(i+1);
   for j=i+1:N

       B(i,j) = prod(b(i+1:j));
       B(j,i) = B(i,j);
   end
end
% Covariance matrix for X (represented as [X_1,...X_M]')
[Gamma]=kron(A,B);
Gamma=sigma2*Gamma;
%% Simulate multivariate normal observations based on Gamma
K = M*N;
Z=randn(K,1);
L=chol(Gamma);
X=L'*Z;
X=reshape(X,[N,M]);
```

grf_miss.m: function to simulate missing observations

```
%% function to generate missing observations for grf_OU
% INPUT:
% X          = complete-observation grf
% miss_leve = % of the observations that is missing
% OUTPUT:
% X_miss     = X with missing values according to miss_level
function X_miss= grf_miss(X,miss_level)
N=size(X,1);
M=size(X,2);
X=reshape(X,[N*M,1]);
% randomly permute the sampling sites:
perm_ind = randperm(N*M);
% Choose the % of observations missing as represented by the index:
miss=perm_ind(1:floor(miss_level*(N*M)));
X_miss=X;
X_miss(miss)=NaN;
X_miss=reshape(X_miss,[N,M]);
```

grf_OU_cond.m : function to simulate $\mathcal{X}|\mathcal{X}^{(o)},\theta^{(p)}$

```
%% Evaluate the conditonal mean and variance for the missing obs in grf_OU
%
% INPUT:
% X        = N-by-M centered Gaussian random (OU) field (with missing observations).
% Gamma    = Covariance structure of the OU field, evaluated at the current parameter value.
%
% OUTPUT:
% X_cond   = Random field where missing observations are replaced with
% conditonal mean.
% Cov_cond = conditional covariance matrix for the unobserved samples
function [X_cond,S_oo,S_uu,S_ou,S_uo]=grf_OU_cond(X,Gamma)

% reshape X into a MN-by-1 vector:
N=size(X,1);
M=size(X,2);
X= reshape(X,[N*M,1]);
% get unobserved and observed indices:
[Unobs_ind] = find(isnan(X)==1);
[Obs_ind]=find(isnan(X)==0);
% Partition the covariance matrix S= [S_oo | S_ou; S_uo | S_uu]:
S_oo = Gamma(Obs_ind,Obs_ind);
```

```
S_uu = Gamma(Unobs_ind, Unobs_ind);
S_ou = Gamma(Obs_ind,Unobs_ind);
S_uo=S_ou';

X_unobs = (S_uo/(S_oo)) * X(Obs_ind);
X_cond=X;
X_cond(Unobs_ind)=X_unobs;
X_cond=reshape(X_cond,[N,M]);
```

lmn_ou2.m : function to evaluate $l(\theta|\mathcal{X})$

```
%% Function to evaluate the likelihood function of the parameter values
    given an observation of the OU process
function likelihood = lmn_ou2(lambda,mu,sigma2,u,v,X,B)
N=size(X,1);
M=size(X,2);
eta=zeros(M,1);
nu=zeros(N,1);
eta(1)=u(1);
nu(1) =v(1);
eta(2:M)=abs(u(2:M)-u(1:M-1));
nu(2:N)=abs(v(2:N)-v(1:N-1));

% l_mn = term_pi+term_sigma+term_lambda+term_mu + double_sum + long_term
    ;
term_pi= M*N*log(2*pi);
term_sigma= log(sigma2);
term_lambda=sum(log(sigma2*(1-exp(-2*lambda*eta(2:M)))));
term_mu = sum(log(sigma2*(1-exp(-2*mu*nu(2:N)))));
double_sum = (M-1)*(N-1)*log(sigma2)...
            +(N-1)*sum(log(1-exp(-2*lambda*eta(2:M))))...
            +(M-1)*sum(log(1-exp(-2*mu*nu(2:N))));
% Define the last long term in the likelihood function:
Xa=zeros(size(X));
long_term=zeros(M,1);
```

```
Xa(:,1)=X(:,1);

for i=2:M
   Xa(:,i)=X(:,i)-(exp(-lambda*eta(i))*X(:,i-1));
   long_term(i)= (Xa(:,i)'/(B)*Xa(:,i))./(1-exp(-2*lambda*eta(i)));
end
likelihood=term_pi+term_sigma+term_lambda+term_mu + double_sum + ...
          (X(:,1)'/(B)*X(:,1)+sum(long_term(2:M)))/sigma2;
```

EM_OU.m : function to implement the EM algorithm

```
%% Function to implement the EM algorithm
function [theta_new,lmb,sig,iter] = EM_OU(X_miss,u,v,lambdap,sigma2p,mup)
%% !! we will keep mu fixed for now!!
% set tolerance and max number of iterations
tol=0.001;
max_step=200;
iter=1;
% Initial parameter values:
theta_p=[lambdap;sigma2p;mup];
theta_new=zeros(3,1);
err=1;
lmb=[];
sig=[];
while err > tol && iter < max_step
    % Evaluate covariance matrix wrt current parameter value:
    [~,~,B,Gamma]=grf_OU(u,v,sigma2p,mup,lambdap);
    % Generate conditional random field:
    [X_cond,~,~,~,~]=grf_OU_cond(X_miss,Gamma);
    % Minimize the likelihood function with current parameter value:
    theta_new(1)=fminsearch(@(lambda) lmn_ou2(lambda,mup,sigma2p,u,v,X_cond,B),1);
    theta_new(2)=fminsearch(@(sigma2) lmn_ou2(lambdap,mup,sigma2,u,v,X_cond,B),1);
```

```
    theta_new(3)=mup;
    theta_p=[lambdap;sigma2p;mup];
    lambdap=theta_new(1);
    sigma2p=theta_new(2);
    % Keep track of error:
    err = (theta_new-theta_p)'*(theta_new-theta_p);
    % Keep track of updates of the parameters:
    lmb(iter)=lambdap;
    sig(iter)=sigma2p;
    % Update iteration:
    iter=iter+1;

end
```

grf_OU_eg1.m : script to run an illustrative example

```
%% An example of realization of an OU GRF:
 clear all
 close all
 M=30;N=30;
 u=linspace(0,1,M);
 v=linspace(0,1,N);
 %u=sort(rand(M,1));
 %v=sort(rand(N,1));
 % Define true parameter values
 sigma2=10; lambda=4;mu=5;
 % Obtain a realization of X
 [X,A,B,Gamma]=grf_OU(u,v,sigma2,mu,lambda);
 % labeling axes for graphs:
 yticklabels=0:0.1:1;
 % Plot the result:
 figure(1)
 imagesc(u,v,X)
 title(['\sigma^2 = ',num2str(sigma2),', \mu = ',num2str(mu),', \lambda
     = ', num2str(lambda)] )
 colorbar;
 set(gca, 'YTickLabel',sort(yticklabels,'descend'));
 %% Simulate a missing-observation grf : 5% data is unobserved
 miss_level=0.05;
 X_miss= grf_miss(X,miss_level);
```

```
% Plot the result:
 figure(2)
 imagesc(u,v,X_miss)
 title(['\sigma^2 = ',num2str(sigma2),', \mu = ',num2str(mu),', \lambda = ',...
     num2str(lambda), ', miss = ',num2str(miss_level)])
 set(gca, 'YTickLabel',sort(yticklabels,'descend'));
%% Implement the EM algorithm:
  tic
  [theta_new,lmb,sig,iter] = EM_OU(X_miss,u,v,1,1,5);
  toc
% Now look at the random field with parameter values estimated from the
% EM algorithm with mu fixed:
  sigma2_new=theta_new(2); lambda_new=theta_new(1);mu_new=mu;
 [~,~,~,Gamma_new]=grf_OU(u,v,sigma2_new,mu_new,lambda_new);
 [X_cond_new,S_oo,S_uu,S_ou,S_uo]=grf_OU_cond(X_miss,Gamma_new);
 figure(3)
 imagesc(u,v,X_cond_new)
  title(['\sigma^2 = ',num2str(sigma2_new),', \mu = ',num2str(mu_new),', \lambda = ', num2str(lambda_new)] )
 colorbar;
 set(gca, 'YTickLabel',sort(yticklabels,'descend'));
 %% Evaluate likelihood function for sigma2:
  sigmap=linspace(1,50,1000);
  likelihood_plot(lambda,mu,sigmap,u,v,X,X_cond_new,B,4)
```

```
%% Evaluate the likelihood function for lambda:
lambdap=linspace(1,50,1000);
likelihood_plot(lambdap,mu,sigma2,u,v,X,X_cond_new,B,5)
mup=linspace(-2,10,100);
likelihood_plot(lambda,mup,sigma2,u,v,X,X_cond_new,B,6)
figure(7)
plot(1:iter-1,lmb,'k-',iter,lmb(iter-1),'ro')
legend('\lambda_p',['\lambda = ',num2str(lmb(end))])
figure(8)
plot(1:iter-1,sig,'k-',iter,sig(iter-1),'ro')
legend('\sigma^2_p',['\sigma^2 = ',num2str(sig(end))])
```