```
function K = MTGP_covMaternisoU_shift(d, hyp, x, z, i)

% Matern covariance function with nu = d/2 and isotropic distance measure. For
% d=1 the function is also known as the exponential covariance function or the
% Ornstein-Uhlenbeck covariance in 1d.
%
% Based on the covMaternisoU.m function of the GPML Toolbox -
%   with the following changes:
%       - only elements of x(:,1:end-1)/z(:,1:end-1) will be analyzed,
%       - x(:,end)/z(:,end) will be ignored, as it contains only the label information
%       - independent of the label all x values will have the same hyp
%       - output scaling hyperparameter is fixed to 1
%       - function considers additional hyperparameter theta_s for features shift
%           (function is limited to 1D features)
%
% The covariance function is:
%
%   k(x^p,x^q) = f( sqrt(d)*r ) * exp(-sqrt(d)*r)
%
% with f(t)=1 for d=1, f(t)=1+t for d=3 and f(t)=1+t+tÂ²/3 for d=5.
% Here r is the distance sqrt(((x-theta_s)^p-(x-theta_s)^q)'*inv(P)*
% ((x-theta_s)^p-(x-theta_s)^q)), P is ell times the unit matrix.
% The hyperparameters are:
%
% hyp = [ log(ell) ]
%           theta_s(1)
%           ...
%           theta_s(nL-1)]
%
% by Robert Duerichen
% 04/02/2014

if nargin<3, K = 'nL'; return; end                  % report number of parameters
if nargin<4, z = []; end                                   % make sure, z exists
xeqz = numel(z)==0; dg = strcmp(z,'diag') && numel(z)>0;        % determine mode

nL = max(x(:,end));
ell = exp(hyp(1));
shift = (hyp(2:end));
if all(d~=[1,3,5]), error('only 1, 3 and 5 allowed for d'), end         % degree

switch d
  case 1, f = @(t) 1;               df = @(t) 1;
  case 3, f = @(t) 1 + t;           df = @(t) t;
  case 5, f = @(t) 1 + t.*(1+t/3);  df = @(t) t.*(1+t)/3;
end
          m = @(t,f) f(t).*exp(-t); dm = @(t,f) df(t).*exp(-t);

%% perform shift
for ii = 2:nL
   x(x(:,end)== ii,1) = x(x(:,end)== ii,1)+shift(ii-1);
   if ~isempty(z)
       z(z(:,2)== ii,1) = z(z(:,2)== ii,1)+shift(ii-1);
   end
end

% precompute distances
if dg                                                               % vector kxx
  K = zeros(size(x(:,1:end-1),1),1);
else
  if xeqz                                                 % symmetric matrix Kxx
    K = sqrt( sq_dist(sqrt(d)*x(:,1)'/ell) );
  else                                                   % cross covariances Kxz
    K = sqrt( sq_dist(sqrt(d)*x(:,1)'/ell,sqrt(d)*z(:,1)'/ell) );
  end
end

if nargin<5                                                        % covariances
  K = m(K,f);
else                                                               % derivatives
  if i==1
```

```
    K = K.*dm(K,f);
  elseif i > 1 && i <= nL
      dim = mod(i,2)+1;
      ind_i = (x(:,2) ==i);
      ind_ni = (x(:,2) ~=i);
      B = zeros(length(x));
      B(ind_ni,ind_i) = ones(sum(ind_ni),sum(ind_i));
      B(ind_i,ind_ni) = -ones(sum(ind_i),sum(ind_ni));
      A = repmat(x(:,dim) ,[1 length(x)]);
      switch d
          case 1
            K = B.*exp(-K)./(ell).*sign(A-A');
          case 3
            K = B.*sqrt(d).*dm(K,f)./(ell).*sign(A-A');
          case 5
            K = sqrt(d).*(K.^2 +K)./(3*ell).*exp(-K).*B.*sign(A-A');

      end
  else
    error('Unknown hyperparameter')
  end
end
```