```
function K = MTGP_covSEisoU_shift_nD(hyp, x, z, i)

% Squared Exponential covariance function with isotropic distance and scaling
% measure which includes a time schift hyperparameter for all signals relative to
% the first signal dimension (which leads to dim-1 additional hyp)
%
% Based on the covSEiso.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) 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
```

$\sigma_f = 1$

```
%       - function considers additional hyperparameter theta_s for features shift
%           (function allows individual shift parameter for each dimension of each
%           task)
%
% The covariance function is parameterized as:
%
% k(x^p,x^q) = exp(-((x-theta_s)^p - ((x-theta_s)^q))'*inv(P)*((x-theta_s)^p - (x-↙
theta_s)^q)/2)
```

$D \times 1$ $D \times D$ $D \times 1$

```
%
% where the P matrix is ell^2 times the unit matrix.
% The hyperparameters are:
%
% hyp = [ log(ell);
%           theta_s(1,1)
%           ...
%           theta_s(1,D)
%           ...
%           theta_s(nL-1,1)
%           ...
%           theta_s(nL-1,D)]
```

$$\theta_s = \begin{bmatrix} \theta_{11} & \theta_{21} & \cdots & \theta_{nL-1,1} \\ \theta_{12} & \theta_{22} & \cdots & \theta_{nL-1,2} \\ \vdots & \vdots & & \vdots \\ \theta_{1D} & \theta_{2D} & \cdots & \theta_{nL-1,D} \end{bmatrix}_{D\times(nL-1)}$$

```
%
% here D is the dimension of the feature
%
% by Robert Duerichen
% 04/02/2014

if nargin<2, K = '1+(D-1)*(nL-1)'; return; end                  % report number of↙
parameters
```

```
if nargin<3, z = []; end                                   % make sure, z exists
xeqz = numel(z)==0; dg = strcmp(z,'diag') && numel(z)>0;       % determine mode

nL = max(x(:,end));                                  % get number of labels
D = size(x,2)-1;                                  % get number of labels
ell = exp(hyp(1));                                 % characteristic length scale
shift = (hyp(2:end));                              % time shift hyp

%% perform shift
for cnt_task = 2:nL
   for cnt_dim = 1:D
        x(x(:,end)== cnt_task,cnt_dim) = x(x(:,end)== cnt_task,cnt_dim)+shift↙
((cnt_task-2)*D+cnt_dim);
        if ~isempty(z)
           z(z(:,end)== cnt_task,cnt_dim) = z(z(:,end)== cnt_task,cnt_dim)+shift↙
((cnt_task-2)*D+cnt_dim);
        end
   end
end

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

if nargin<4                                                       % covariances
  K = exp(-K/2);
else                                                              % derivatives
  if i<=1+(D)*(nL-1)
      if i == 1 % derivative of the x-scaling hyperparameter
        K = exp(-K/2).*K;
      else  % derivatives of the shift hyperparameters
          task = fix((i)/D)+1;
          dim = mod(i,2)+1;
          ind_i = (x(:,end) ==task);
          ind_ni = (x(:,end) ~=task);
          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)]);
          K = B.*((A-A’)./(ellˆ2).*exp(-K/2));
      end
  else
    error(’Unknown hyperparameter’)
  end
end
```