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

% Squared Exponential covariance function with different distance
% measure for each task.
%
% Based on the covSEisoU.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) contains the label information
%       - tash specific input-scaling hyperparameter based on convolution
%       - output-scaling hyperparameter is fixed to 1
%
% The covariance function is parameterized for a two inputs x_1 and x_2 from
% task l=1 and l=2 as:
%
% k(x_1,x_2) = sqrt(2*ell_1*ell_2/(ell_1^2*ell_2^2))*exp(-(x_1 -
%                       x_2)^2/(ell_1^2+ell_2^2));
%
% The hyperparameters are:
%
% hyp = [   log(ell_1)
%           ...
%           log(ell_nL)]
%
%
% by Robert Duerichen
% 01/21/2014


if nargin<2, K = 'nL'; return; end                                  % report number of
parameters
                                                                    % here nL is number of
                                                                    tasks
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(:,2));                                                   % determine dimension

ell = exp(hyp(1:nL));                                               % characteristic length
scale


if dg                                                               % vector kxx
    K = zeros(size(x(:,1),1),1);                                    % initialise correlation
    vector
    r = zeros(size(x(:,1),1),1);                                    % initialise squared
    distance vector
    hyp_mat = ones(size(x(:,1),1),1,4);                             % initialise 4D matrix
    to store hyperparameter results
else
    if xeqz                                                         % symmetric matrix Kxx

        r = sq_dist(x(:,1)');                                       % compute squared distance

        len_x = length(x(:,1));
        hyp_mat = zeros(len_x,len_x,4);                             % initialise 4D matrix
        to store hyperparameter results
        hyp_mat(:,:,1) = repmat(ell(x(:,2))',[1,len_x]);            % repeat hyperparameter
        in dim 2 of matrix
        hyp_mat(:,:,2) = repmat(ell(x(:,2)),[len_x,1]);             % repeat hyperparameter
        in dim 1 of matrix

    else                                                            % cross covariances Kxz
        r = sq_dist(x(:,1)',z(:,1)');                               % compute squared distance

        len_x = length(x(:,1));
        len_z = length(z(:,1));
        hyp_mat = zeros(len_x,len_z,4);                             % initialise 4D matrix
        to store hyperparameter results
        hyp_mat(:,:,1) = repmat(ell(x(:,2))',[1,len_z]);            % repeat hyperparameter
        in dim 2 of matrix
```

```
61          hyp_mat(:,:,2) = repmat(ell(z(:,2)),[len_x,1]);     % repeat hyperparameter
            in dim 1 of matrix
62      end
63
64      hyp_mat(:,:,3) = (hyp_mat(:,:,1).^2+hyp_mat(:,:,2).^2); % comp. squared sum of
        hyperpara.
65      hyp_mat(:,:,4) = sqrt(2*hyp_mat(:,:,1).*hyp_mat(:,:,2)./(hyp_mat(:,:,3)));
66
67      K = r./(hyp_mat(:,:,3));                                % comp. correlation matrix
68  end
69
70  if nargin<4                                                 % covariances
71      K = hyp_mat(:,:,4).* exp(-K);
72  else                                                        % derivatives
73      if i <= nL
74          % case hyper ell(i) does not belong to x_1 and x_2
75          K(x(:,2) ~= i,x(:,2) ~= i) = 0;
76
77          % case x_1 and x_2 belong to do not belong to the same task (l_1~=l_2 -->
            ell_1~=ell_2)
78          K(x(:,2) ~= i,x(:,2) == i) = hyp_mat(x(:,2) ~= i,x(:,2) == i,1)
            .*hyp_mat(x(:,2) ~= i,x(:,2) == i,2) .* exp(-K(x(:,2) ~= i,x(:,2) ==
            i)).*(4.*r(x(:,2) ~= i,x(:,2) == i).*hyp_mat(x(:,2) ~= i,x(:,2) ==
            i,1).^2-hyp_mat(x(:,2) ~= i,x(:,2) == i,1).^4+hyp_mat(x(:,2) ~= i,x(:,2) ==
            i,2).^4) ./ ...
79              (hyp_mat(x(:,2) ~= i,x(:,2) == i,4).*hyp_mat(x(:,2) ~= i,x(:,2) ==
                i,3).^3);
80
81          K(x(:,2) == i,x(:,2) ~= i) = K(x(:,2) ~= i,x(:,2) == i)';
82
83          % case x_1 and x_2 belong to same task (l_1=l_2 --> ell_1=ell_2)
84          K(x(:,2) == i,x(:,2) == i) = hyp_mat(x(:,2) == i,x(:,2) == i,1) .*
            exp(-K(x(:,2) == i,x(:,2) == i)).*r(x(:,2) == i,x(:,2) == i) ./ ...
85              (hyp_mat(x(:,2) == i,x(:,2) == i,1).^3);
86      else
87          error('Unknown hyperparameter')
88      end
89  end
```