```matlab
function [K] = MTGP_covQPSisoUU_shift_mask(mask,hyp, x, z, i)

% Stationary covariance function for a quasi-periodic function based on a
% multiplication of a Matern and Periodic function
%
%       - 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
%       - feature scaling hyperparameter is fixed to 1
%       - output scaling hyperparameter is fixed to 1
%       - mask parameter is a vector of size hyp and if mask(i) == 0, the
%       derivative of hyp(i) will be 0
%
% k(x,y) = exp(-((x-theta_s) - ((y-theta_s)))'*inv(P)*((x-theta_s) - (y-theta_s)^q)/2)  * ...
%               exp( -2*sin^2( pi*||(x-theta_s)-(y-theta_s)||/p ) )
%
% where the P matrix is ell^2 times the unit matrix.
% The hyperparameters are:
%
% hyp = [ log(ell)
%         log(p);
%         theta_s(1)
%           ...
%         theta_s(nL-1)]
%
% modified by Robert Duerichen
% 8/11/2013
%
% See also COVFUNCTIONS.M.

if nargin<3, K = 'nL+1'; 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

% check if size of mask is correct
if size(mask) ~= size(hyp)
    error('Size of mask vector is not equivalent to hyperparameter vector');
end

% check if derivate shall be computed or not
if exist('i','var')
    if mask(i) == 0
        if xeqz                                    % symmetric matrix Kxx
            K = zeros(length(x));
        else                                       % cross covariances Kxz
            K = zeros(length(x),length(z));
        end
        return;                                      % terminate function
    end
end

nL = max(x(:,2));                                  % get number of labels
ell = exp(hyp(1));                                 % characteristic length scale
p   = exp(hyp(2));                                 % period
shift = (hyp(3:end));                              % time shift hyp

%% perform shift
for ii = 2:nL
   x(x(:,2)== ii,1) = x(x(:,2)== 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_p = zeros(size(x(:,1),1),1);
  K_se = zeros(size(x(:,1),1),1);
```

```
else
  if xeqz                                                 % symmetric matrix Kxx
    K_se = sq_dist(x(:,1:end-1)'/ell);
    K_p = sqrt(sq_dist(x(:,1:end-1)'));
  else                                                   % cross covariances Kxz
    K_se = sq_dist(x(:,1:end-1)'/ell,z(:,1:end-1)'/ell);
    K_p = sqrt(sq_dist(x(:,1:end-1)',z(:,1:end-1)'));
  end
end

K_p = pi*K_p/p;
if nargin<5                                                        % covariances
    K_p = sin(K_p); K_p = K_p.*K_p; K_p =   exp(-2*K_p);
    K_se = exp(-K_se/2);
    K = K_p.*K_se;
else                                                               % derivatives
    if i<=nL+1
        if i==1         % derivatives of the se hyperparameter
            K_p = sin(K_p); K_p = K_p.*K_p; K_p =   exp(-2*K_p);
            K = K_p.*exp(-K_se/2).*K_se;
        elseif i==2         % derivatives of the periodic hyperparameter
            K_se = exp(-K_se/2);
            R = sin(K_p); K = K_se.* 4.*exp(-2*R.*R).*R.*cos(K_p).*K_p;
        else  % derivatives of the shift hyperparameters
          ind_i = (x(:,2) ==i-1);
          ind_ni = (x(:,2) ~=i-1);
          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(:,1) ,[1 length(x)]);

          R = sin(K_p);
          dK_p = B.*4.*exp(-2*R.*R).*R.*cos(K_p).*pi./p.*sign(A-A');

          dK_se = B.*((A-A')./(ell^2).*exp(-K_se/2));

          K_p = sin(K_p); K_p = K_p.*K_p; K_p =   exp(-2*K_p);
          K_se = exp(-K_se/2);

          K = dK_se.*K_p + K_se.*dK_p;
      end
    else
        error('Unknown hyperparameter')
    end

end
```