```matlab
function K = MTGP_covADD(cov, hyp, x, z, i)

% Additive covariance function using a 1d base covariance function
% cov(x^p,x^q;hyp) with individual hyperparameters hyp.
%
% k(x^p,x^q) = \sum_{r \in R} sf_r \sum_{|I|=r}
%                  \prod_{i \in I} cov(x^p_i,x^q_i;hyp_i)
%
% hyp = [ hyp_1
%         hyp_2
%          ...
%         hyp_D
%         log(sf_R(1))
%          ...
%         log(sf_R(end)) ]
%
% where hyp_d are the parameters of the 1d covariance function which are shared
% over the different values of R(1) to R(end).
%
%
% Copyright (c) by Carl Edward Rasmussen and Hannes Nickisch, 2010-09-10.
%
% See also COVFUNCTIONS.M.

R = cov{1};
nh = eval(feval(cov{2}));           % number of hypers per individual covariance
nr = numel(R);                      % number of different degrees of interaction
if nargin<3                                  % report number of hyper parameters
  K = ['D*', int2str(nh), '+', int2str(nr)];
  return
end
if nargin<4, z = []; end                                   % make sure, z exists
xeqz = isempty(z); dg = strcmp(z,'diag');                        % determine mode

[n,D] = size(x(:,end-1));                                               % dimensionality x-->x(:,end-1)
sf2 = exp( 2*hyp(D*nh+(1:nr)) );         % signal variances of individual degrees
%these lines have to be added to be able to use Lab_covCC_chol_nD function
if size(x,2) > 1
    nL = max(x(:,end));
end
Kd = Kdim(cov{2},hyp,x(:,end-1),z);                 % evaluate dimensionwise covariances K
if nargin<5                                                         % covariances
  EE = elsympol(Kd,max(R));                  % Rth elementary symmetric polynomials
  K = 0; for ii=1:nr, K = K + sf2(ii)*EE(:,:,R(ii)+1); end    % sf2 weighted sum
else                                                                % derivatives
  if i <= D*nh                         % individual covariance function parameters
    j = fix(1+(i-1)/nh);                 % j is the dimension of the hyperparameter
    if dg, zj='diag'; else if xeqz, zj=[]; else zj=z(:,j); end, end
    dKj = feval(cov{2},hyp(nh*(j-1)+(1:nh)),x(:,j),zj,i-(j-1)*nh);  % other dK=0
    % the final derivative is a sum of multilinear terms, so if only one term
    % depends on the hyperparameter under consideration, we can factorise it
    % out and compute the sum with one degree less
    E = elsympol(Kd(:,:,[1:j-1,j+1:D]),max(R)-1);  %  R-1th elementary sym polyn
    K = 0; for ii=1:nr, K = K + sf2(ii)*E(:,:,R(ii)); end      % sf2 weighted sum
    K = dKj.*K;
  elseif i <= D*nh+nr
    EE = elsympol(Kd,max(R));                % Rth elementary symmetric polynomials
    j = i-D*nh;
    K = 2*sf2(j)*EE(:,:,R(j)+1);                       % rest of the sf2 weighted sum
  else
    error('Unknown hyperparameter')
  end
end

% evaluate dimensionwise covariances K
function K = Kdim(cov,hyp,x,z)
  [n,D] = size(x);                                                 % dimensionality
  nh = eval(feval(cov));            % number of hypers per individual covariance
  if nargin<4, z = []; end                                 % make sure, z exists
```

```
  xeqz = numel(z)==0; dg = strcmp(z,'diag') && numel(z)>0;      % determine mode

  if dg                                                            % allocate memory
    K = zeros(n,1,D);
  else
    if xeqz, K = zeros(n,n,D); else K = zeros(n,size(z,1),D); end
  end

  for d=1:D
    hyp_d = hyp(nh*(d-1)+(1:nh));                      % hyperparamter of dimension d
    if dg
      K(:,:,d) = feval(cov,hyp_d,x(:,d),'diag');
    else
      if xeqz
        K(:,:,d) = feval(cov,hyp_d,x(:,d));
      else
        K(:,:,d) = feval(cov,hyp_d,x(:,d),z(:,d));
      end
    end
  end
```