```
function K = MTGP_covMaterniso(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 covMaterniso.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
%
%The covariance function is:
%
%   k(x^p,x^q) = s2f * 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^p-x^q)'*inv(P)*(x^p-x^q)), P is ell times
% the unit matrix and sf2 is the signal variance. The hyperparameters are:
%
% hyp = [ log(ell)
%         log(sqrt(sf2)) ]
%
% modified by Robert Duerichen
% 04/02/2014

if nargin<3, K = '2'; 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

ell = exp(hyp(1));
sf2 = exp(2*hyp(2));
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).*t.*exp(-t);

% 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:end-1)'/ell) );
  else                                                % cross covariances Kxz
    K = sqrt( sq_dist(sqrt(d)*x(:,1:end-1)'/ell,sqrt(d)*z(:,1:end-1)'/ell) );
  end
end

if nargin<5                                                       % covariances
  K = sf2*m(K,f);
else                                                              % derivatives
  if i==1
    K = sf2*dm(K,f);
  elseif i==2
    K = 2*sf2*m(K,f);
  else
    error('Unknown hyperparameter')
  end
end
```