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

% Rational Quadratic covariance function with isotropic distance measure.
%
% Based on the covRQiso.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
% The covariance function is parameterized as:
%
% k(x^p,x^q) = sf2 * [1 + (x^p - x^q)'*inv(P)*(x^p - x^q)/(2*alpha)]^(-alpha)
%
% where the P matrix is ell^2 times the unit matrix, sf2 is the signal
% variance and alpha is the shape parameter for the RQ covariance. The
% hyperparameters are:
%
% hyp = [ log(ell)
%         log(sqrt(sf2))
%         log(alpha) ]
%
% by Robert Duerichen
% 04/02/2014

if nargin<2, K = '3'; 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

ell = exp(hyp(1));
sf2 = exp(2*hyp(2));
alpha = exp(hyp(3));

% precompute squared distances
if dg                                                               % vector kxx
  D2 = zeros(size(x(:,end-1),1),1);
else
  if xeqz                                                % symmetric matrix Kxx
    D2 = sq_dist(x(:,end-1)'/ell);
  else                                                  % cross covariances Kxz
    D2 = sq_dist(x(:,end-1)'/ell,z(:,end-1)'/ell);
  end
end

if nargin<4                                                         % covariances
  K = sf2*((1+0.5*D2/alpha).^(-alpha));
else                                                                % derivatives
  if i==1                                              % length scale parameter
    K = sf2*(1+0.5*D2/alpha).^(-alpha-1).*D2;
  elseif i==2                                           % magnitude parameter
    K = 2*sf2*((1+0.5*D2/alpha).^(-alpha));
  elseif i==3
    K = (1+0.5*D2/alpha);
    K = sf2*K.^(-alpha).*(0.5*D2./K - alpha*log(K));
  else
    error('Unknown hyperparameter')
  end
end
```