```
function K = MTGP_covCC_chol_nD_mask(mask,hyp,x, z, i)
%
% Generates a "free-form" cross correlation covariance matrix as proposed
% by Bonilla et al. 2007 using a Cholesky decomposition to assure that the
% matrix is positive definite
%
% hyperparameters are the elements of the lower triangula matrix L in the order
% of:
%           theta_c,1       0               0                       0
%   L   = [ theta_c,2       theta_3         0               ...     0           ]
%               ...
%           theta_c,k-m+1   theta_c,k-m+2   theta_c,k-m+3   ...    theta_c,k
%
% Parametrization is as discribed "Tutorial on Multi-Task Gaussian
% Processes for biomedical applications"
%
% Only elements of x(:,end)/z(:,end) will be analyzed, residual columns will be ignored.
% x(:,end)/z(:,end) contain the label information of the feature
%
% Derivatives are implemented and hypperparameters can be optimized via gradient descent
% (So far only tested up to nL 4)
%
%
% hyp = [    (theta_c,1)
%            (theta_c,2)
%            ...
%            (theta_c,k)]
%
%       - mask parameter is a vector of size hyp and if mask(i) == 0, the
%       derivative of hyp(i) will be 0
%
% by Robert Duerichen
% 04/02/2014

if nargin<3, K = ['sum([1:nL])']; 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(:,end));                                  % determine nLension
cc = (hyp(1:sum([1:nL])));                                     % ini

% create index for hyp in matrix L
cnt =1;
for cnt_nL1 = 1:nL
    for cnt_nL2 = 1:cnt_nL1
        ind_cc(cnt,:) = [cnt_nL1,cnt_nL2];
        cnt = cnt+1;
    end
end

% compute K_f
L = zeros(nL,nL);
```

```matlab
for cnt_ind = 1:size(ind_cc,1)
    L(ind_cc(cnt_ind,1),ind_cc(cnt_ind,2)) = cc(cnt_ind);
end
K_f = L*L';

% precompute squared distances
if nargin<5
    if dg                                                               % vector kxx
        K = corr_nd(x(:,end), x(:,end),K_f);
        K = diag(K);
    else
        if xeqz                                                 % symmetric matrix Kxx
            K = corr_nd(x(:,end), x(:,end),K_f);
        else                                                   % cross covariances Kxz
            K = corr_nd(x(:,end), z(:,end),K_f);
        end
    end

else % derivatives
    dL = zeros(nL,nL);
    if i <= length(ind_cc)
        dL(ind_cc(i,1),ind_cc(i,2)) = 1;
    else
        K = 0;
%         error('Unknown hyperparameter')
    end


    dK_f = dL*L' + L*dL';

    if xeqz                                                 % symmetric matrix Kxx
        K = corr_nd(x(:,end), x(:,end),dK_f);
    else                                                   % cross covariances Kxz
        K = corr_nd(x(:,end), z(:,end),dK_f);
    end
end
```