```matlab
function [U,S,V,SOBJ,ErrFlag] = MLPCA(X,stdX,p);
%
%                 MLPCA.M   v. 4.0
%
% This function performs maximum likelihood principal components
% analysis assuming uncorrelated measurement errors.  MLPCA is a
% method that attempts to provide an optimal estimation of the p-
% dimensional subspace containing the data by taking into account
% uncertainties in the measurements, thereby dealing with those
% cases that cannot be treated by simple scaling.
%
% The variables % passed to the function are:
%
%   X    is the mxn matrix of observations (measurements).
%
%   stdX is the mxn matrix of standard deviations associated with
%        the observations in X.  For missing measurements, stdX
%        should be set to zero.
%
%   p    is the dimensionality of the subspace sought
%        (i.e. the pseudo-rank)  (p < n and m).
%
% The parameters returned are:
%
%   U,S,V  are pseudo-svd parameters (mxp, pxp, and nxp).  The
%          maximum likelihood estimates are given by:
%                   XML=U*S*V'
%
%   SOBJ is the value of the objective function for the best model.
%        For exact uncertainty estimates, this should follow a
%        chi-squared distribution with (m-p)*(n-p) degrees of
%        freedom.
%
%   ErrFlag indicates the termination conditions of the function;
%             0 = normal termination (convergence)
%             1 = maximum number of iterations exceeded
%
% The function can also produce a file - mlpca.mat - if appropriate
% lines are activated as indicated in the code.  This can be used to
% follow convergence if desired.
%
% Similarities to PCA:
%   - the columns of U are orthonormal; the columns of V are
%     orthonormal; S is diagonal.
%   - U*S gives the maximum likelihood scores (which can be used
%     for PCR
%
% Differences from PCA:
%   - Solutions are not nested - i.e. the rank (p+1) solution does
%     not have the rank p solution as a subset.  Therefore, the rank
%     of the subspace sought needs to be specified in advance.
%   - Unlike PCA, MLPCA uses maximum likelihood projection rather
%     an orthogonal projection to estimate new points in the subspace.
%     The ML projection is weighted by the errors.  For example, if
%     U,S, and V are the MLPCA results from the decomposition of X
%     which is mxn, then the score vector for the projection a new
```

```
%     vector of measurements, x (nx1), is,
%
%               t = inv(V'*Q*V)*V'*Q*x
%
%     where t is the (px1) vector of scores and Q is the inverse of
%     (nxn) diagonal matrix of measurement variances.  A similar
%     equation gives the maximum likelihood estimate in the original
%     space:
%
%               xml = V*t
%
%     Note that these reduce to the normal orthogonal projections
%     (t=V*x, xml=V'*V*x) when all of the measurement uncertainties
%     are equal.  It is essential to do ML projections rather than
%     orthogonal projections with MLPCA, since the latter counteract
%     the advantages of the ML decomposition.
%   - Mean centering can be used prior to MLPCA, but technically
%     this invalidates the "maximum likelihood" features to a greater
%     or lesser extent, since these quantities are not estimated in
%     an ML fashion.  (Work is ongoing on a variation of the algorithm
%     to handle this case.)
%   - Scaling prior to MLPCA is superfluous, since it is intended
%     to eliminate that necessity.
%
% Initialization
%
convlim=1e-10;             % convergence limit
maxiter=50;             % maximum no. of iterations
XX=X;                      % XX is used for calculations
varX=(stdX.^2);            % convert s.d.'s to variances
[i,j] = find(varX==0);     % find zero errors and convert to large
errmax = max(max(varX));   % errors for missing data
for k=1:length(i);
   varX(i(k),j(k)) = 1e+10*errmax;
end
n=length(XX(1,:));         % the number of columns
%
% Generate initial estimates assuming homoscedastic errors.
%
[U,S,V]=svd(XX,0);         % Decompose adjusted matrix
U0=U(:,1:p);               % Truncate solution to rank p
count=0;                   % Loop counter
Sold=0;                    % Holds last value of objective function
ErrFlag=-1;                % Loop flag
%
% Loop for alternating regression
%
while ErrFlag<0;
   count=count+1;          % Increment loop counter
%
% Evaluate objective function
%
   Sobj=0;                             % Initialize sum
   MLX=zeros(size(XX));                % and maximum likelihood estimates
   for i=1:n                            % Loop for each column of XX
      Q=sparse(diag(varX(:,i).^(-1)));  % Inverse of err. cov. matrix
      F=inv(U0'*Q*U0);                    % Intermediate calculation
      MLX(:,i)=U0*(F*(U0'*(Q*XX(:,i))));  % Max. likelihood estimates
```

```
      dx=XX(:,i)-MLX(:,i);                % Residual vector
      Sobj=Sobj+dx'*Q*dx;                 % Update objective function
   end
%
% This section for diagnostics only and can be commented out.  "Ssave"
% can be plotted to follow convergence.
%
%   Ssave(count)=Sobj;
%   save mlpca;
%
% End diagnostics
%
% Check for convergence or excessive iterations
%
   if rem(count,2)==1                   % Check on odd iterations only
      if (abs(Sold-Sobj)/Sobj)<convlim  % Convergence criterion
         ErrFlag=0;
      elseif count>maxiter              % Excessive iterations?
         ErrFlag=1;
      end
   end
%
% Now flip matrices for alternating regression
%
   if ErrFlag<0                  % Only do this part if not done
      Sold=Sobj;                 % Save most recent obj. function
      [U,S,V]=svd(MLX,0);        % Decompose ML values
      XX=XX';                           % Flip matrix
      varX=varX';                       % and the variances
      n=length(XX(1,:));                % Adjust no. of columns
      U0=V(:,1:p);                      % V becomes U in for transpose
   end
end
%
% All done.  Clean up and go home.
%
[U,S,V]=svd(MLX,0);
U=U(:,1:p);
S=S(1:p,1:p);
V=V(:,1:p);
SOBJ=Sobj;
```

```
Not enough input arguments.

Error in MLPCA (line 85)
XX=X;                      % XX is used for calculations
```

Published with MATLAB® R2023a