```
function [fit, vaf] = arobfit(prox, inperm)

% AROBFIT fits an anti-Robinson matrix using iterative projection to
% a symmetric proximity matrix in the $L_{2}$-norm.
% PROX is the input proximity matrix ($n \times n$ with a zero main diagonal
% and a dissimilarity interpretation);
% INPERM is a given permutation of the first $n$ integers;
% FIT is the least-squares optimal matrix (with variance-accounted-for
% of VAF) to PROX having an anti-Robinson form for the row and column
% object ordering given by INPERM.

function [fit, vaf, outperm] = arobfnd(prox, inperm, kblock)

% AROBFND fits an anti-Robinson matrix using iterative projection to
% a symmetric proximity matrix in the $L_{2}$-norm based on a permutation
% identified through the use of iterative quadratic assignment.
% PROX is the input proximity matrix ($n \times n$ with a zero main diagonal
% and a dissimilarity interpretation);
% INPERM is a given starting permutation of the first $n$ integers;
% FIT is the least-squares optimal matrix (with variance-accounted-for
% of VAF) to PROX having an anti-Robinson form for the row and column
% object ordering given by the ending permutation OUTPERM. KBLOCK
% defines the block size in the use the iterative quadratic assignment
% routine.

function [find,vaf] = atreectul(prox,inperm)

% ATREEFINDCTUL finds and fits an additive tree by first fitting
% a centroid metric (using centfit.m) and secondly an ultrametric to the resudual
% matrix (using ultrafnd.m).
% PROX is the input proximity matrix (with a zero main diagonal
% and a dissimilarity interpretation);
% INPERM is a permutation that determines the order in which the
% inequality constraints are considered;
% FIND is the found least-squares matrix (with variance-accounted-for
% of VAF) to PROX satisfying the additive tree constraints.

function [ulmetric,ctmetric] = atreedec(prox,constant)

% ATREEDEC decomposes a given additive tree matrix into an ultrametric and a
% centroid metric matrix (where the root is half-way along the longest path).
% PROX is the input proximity matrix (with a zero main diagonal
% and a dissimilarity interpretation);
% CONSTANT is a nonnegative number (less than or equal to the maximum
% proximity value) that controls the positivity of the constructed ultrametric values;
% ULMETRIC is the ultrametric component of the decomposition;
% CTMETRIC is the centoid metric component of the decomposition (given
% by values $g_{1},...,g_{n}$ for each of the objects, some of which
% may actually be negative depending on the input proximity matrix used).

function [fit,vaf] = atreefit(prox,targ)

% ATREEFIT fits a given additive tree using iterative projection to
% a symmetric proximity matrix in the $L_{2}$-norm.
% PROX is the input proximity matrix (with a zero main diagonal
% and a dissimilarity interpretation);
% TARG is an matrix of the same size as PROX with entries
% satisfying the four-point additive tree constraints;
% FIT is the least-squares optimal matrix (with variance-accounted-for
% of VAF) to PROX satisfying the additive tree constraints implicit in TARG.

function [find,vaf] = atreefnd(prox,inperm)

% ATREEFND finds and fits an additive tree using iterative projection
% heuristically on a symmetric proximity matrix in the $L_{2}$-norm.
% PROX is the input proximity matrix (with a zero main diagonal
% and a dissimilarity interpretation);
% INPERM is a permutation that determines the order in which the
% inequality constraints are considered;
% FIND is the found least-squares matrix (with variance-accounted-for
% of VAF) to PROX satisfying the additive tree constraints.

function [find,vaf,ultrafit,lengths] = atreefndtm(proxtm,inpermrow,inpermcol)

% ATREEFNDTM finds and fits a two-mode additive tree; iterative projection is used
% heuristically to find a two-mode ultrametric component that
% is added to a two-mode centroid metric to produce the two-mode additive tree.
% PROXTM is the input proximity matrix (with a dissimilarity interpretation);
% INPERMROW and INPERMCOL are permutations for the row and column
% objects that determine the order in which the
% inequality constraints are considered;
% FIND is the found least-squares matrix (with variance-accounted-for
% of VAF) to PROXTM satisfying the additive tree constraints
% the vector LENGTHS contains the row followed by column values for the
% two-mode centroid metric component; ULTRA is the ultrametric component.

function [find,vaf,targone,targtwo,outpermone,outpermtwo] = biarobfnd(prox,inperm,kblock)

% BIAROBFND fits the sum of two anti-Robinson matrices using iterative projection to
% a symmetric proximity matrix in the $L_{2}$-norm based on permutations
% identified through the use of iterative quadratic assignment.
% PROX is the input proximity matrix ($n \times n$ with a zero main diagonal
% and a dissimilarity interpretation);
% INPERM is a given starting permutation of the first $n$ integers;
% FIND is the least-squares optimal matrix (with variance-accounted-for
% of VAF) to PROX and is the sum of the two anti-Robinson matrices
% TARGONE and TARGTWO based on the two row and column
% object orderings given by the ending permutations OUTPERMONE
% and OUTPERMTWO. KBLOCK defines the block size in the use the
% iterative quadratic assignment routine.

function [find,vaf,targone,targtwo] = biatreefnd(prox,inperm)

% BIATREEFND finds and fits the sum of two additive trees using iterative projection
% heuristically on a symmetric proximity matrix in the $L_{2}$-norm.
% PROX is the input proximity matrix (with a zero main diagonal
% and a dissimilarity interpretation);
% INPERM is a permutation that determines the order in which the
% inequality constraints are considered;
% FIND is the found least-squares matrix (with variance-accounted-for
% of VAF) to PROX and is the sum of the two additive tree matrices TARGONE and TARGTWO.
```

```
function [find,vaf,targone,targtwo,outpermone,outpermtwo, addconone, addcontwo] =
bicirac(prox,inperm,kblock)

% BICIRAC finds and fits the sum of two circular unidimensional scales using iterative projection
to
% a symmetric proximity matrix in the $L_{2}$-norm based on permutations
% identified through the use of iterative quadratic assignment.
% PROX is the input proximity matrix ($n \times n$ with a zero main diagonal
% and a dissimilarity interpretation);
% INPERM is a given starting permutation of the first $n$ integers;
% FIND is the least-squares optimal matrix (with variance-accounted-for
% of VAF) to PROX and is the sum of the two circular anti-Robinson matrices
% TARGONE and TARGTWO based on the two row and column
% object orderings given by the ending permutations OUTPERMONE
% and OUTPERMTWO. KBLOCK defines the block size in the use the
% iterative quadratic assignment routine and ADDCONONE and ADDCONTWO are
% the two additive constants for the two model components.

function [find,vaf,targone,targtwo,outpermone,outpermtwo] = bicirarobfnd(prox,inperm,kblock)

% BICIRAROBFND finds and fits the sum of two circular anti-Robinson scales using iterative
projection to
% a symmetric proximity matrix in the $L_{2}$-norm based on permutations
% identified through the use of iterative quadratic assignment.
% PROX is the input proximity matrix ($n \times n$ with a zero main diagonal
% and a dissimilarity interpretation);
% INPERM is a given starting permutation of the first $n$ integers;
% FIND is the least-squares optimal matrix (with variance-accounted-for
% of VAF) to PROX and is the sum of the two circular anti-Robinson matrices
% TARGONE and TARGTWO based on the two row and column
% object orderings given by the ending permutations OUTPERMONE
% and OUTPERMTWO.

function [find,vaf,targone,targtwo,outpermone,outpermtwo] = bicirsarobfnd(prox,inperm,kblock)

% BICIRSAROBFND fits the sum of two stongly circular-anti-Robinson matrices using iterative
% projection to a symmetric proximity matrix in the $L_{2}$-norm based on permutations
% identified through the use of iterative quadratic assignment.
% PROX is the input proximity matrix ($n \times n$ with a zero main diagonal
% and a dissimilarity interpretation);
% INPERM is a given starting permutation of the first $n$ integers;
% FIND is the least-squares optimal matrix (with variance-accounted-for
% of VAF) to PROX and is the sum of the two strongly circular-anti-Robinson matrices
% TARGONE and TARGTWO based on the two row and column
% object orderings given by the ending permutations OUTPERMONE
% and OUTPERMTWO. KBLOCK defines the block size in the use the
% iterative quadratic assignment routine.

function [outpermone,outpermtwo,coordone,coordtwo,fitone,fittwo,addconone, ...
   addcontwo,vaf,monprox] = ...
bimonscalqa(prox,targone,targtwo,inpermone,inpermtwo,kblock,nopt)

%BIMONCALQA carries out a bidimensional scaling of a symmetric proximity
%  matrix using iterative quadratic assignment, plus it provides an
%  optimal monotonic transformation (MONPROX) of the original input
%  proximity matrix.
%  PROX is the input proximity matrix (with a zero main diagonal and a
%  dissimilarity interpretation);
%  TARGONE is the input target matrix for the first dimension (usually with
%  a zero main diagonal and with a dissimilarity interpretation representing
%  equally-spaced locations along a continuum); TARGTWO is the input target
%  matrix for the second dimension;
%  INPERMONE is the input beginning permutation for the first dimension
%  (a permuation of the first $n$ integers); INPERMTWO is the input beginning
%  permutation for the second dimension;
%  the insertion and rotation routines use from 1 to KBLOCK
%  (which is less than or equal to $n-1$) consecutive objects in
%  the permutation defining the row and column orders of the data matrix;
%  NOPT controls the confirmatory or exploratory fitting of the unidimensional
%  scales; a value of NOPT = 0 will fit in a confirmatory manner the two scales
%  indicated by INPERMONE and INPERMTWO; a value of NOPT = 1 uses iterative QA
%  to locate the better permutations to fit;
%  OUTPERMONE is the final object permutation for the first dimension;
%  OUTPERMTWO is the final object permutation for the second dimension;
%  COORDONE is the set of first dimension coordinates in ascending order;
%  COORDTWO is the set of second dimension coordinates in ascending order;
%  ADDCONONE is the additive constant for the first dimensional model;
%  ADDCONTWO is the additive constant for the second dimensional model;
%  VAF is the variance-accounted-for in MONPROX by the bidimensional scaling.

function [find,vaf,targone,targtwo,outpermone,outpermtwo, ...
        rowpermone,colpermone,rowpermtwo,colpermtwo,addconone,...
        addcontwo,coordone,coordtwo,axes,monproxtm]  = …
bimonscaltmac(proxtm,inpermone,inpermtwo,kblock)

% BIMONSCALTMAC finds and fits the sum of two linear unidimensional scales using iterative
projection to
% a two-mode proximity matrix in the $L_{2}$-norm based on permutations
% identified through the use of iterative quadratic assignment.  It also
% provides an optimal monotonic transformation (MONPROX) of the original
% input proximity matrix.
% PROXTM is the input two-mode proximity matrix ($nrow \times ncol$)
% and a dissimilarity interpretation);
% FIND is the least-squares optimal matrix (with variance-accounted-for
% of VAF) to the monotonic transformation MONPROXTM of
% the input proximity matrix and is the sum of the two  matrices
% TARGONE and TARGTWO based on the two row and column
% object orderings given by the ending permutations OUTPERMONE
% and OUTPERMTWO, and in turn ROWPERMONE and ROWPERMTWO and COLPERMONE
% and COLPERMTWO. KBLOCK defines the block size in the use the
% iterative quadratic assignment routine and ADDCONONE and ADDCONTWO are
% the two additive constants for the two model components; The $n$ coordinates
% are in COORDONE and COORDTWO.  The input permutations are INPERMONE and
% INPERMTWO.  The $n \times 2$ matrix AXES gives the plotting coordinates for the
% combined row and column object set.

function [] = biplottm(axes,nrow,ncol)

%BIPLOTTM plots the combined row and column object set using coordinates
% given in the $n \times 2$ matrix AXES; here the number of rows is
% NROW and the number of columns is NCOL, and $n$ is the sum of NROW and
% NCOL.  The first NROW rows of AXES give the row object coordinates;
% the last NCOL rows of AXES give the column object coordinates.  The
% plotting symbol for rows is a circle (o); for columns it is an asterisk (*).
% The labels for rows are from 1 to NROW; those for columns are from 1 to NCOL.

function [find,vaf,targone,targtwo,outpermone,outpermtwo] = bisarobfnd(prox,inperm,kblock)

% BISAROBFND fits the sum of two stongly anti-Robinson matrices using iterative
% projection to a symmetric proximity matrix in the $L_{2}$-norm based on permutations
% identified through the use of iterative quadratic assignment.
% PROX is the input proximity matrix ($n \times n$ with a zero main diagonal
% and a dissimilarity interpretation);
% INPERM is a given starting permutation of the first $n$ integers;
% FIND is the least-squares optimal matrix (with variance-accounted-for
% of VAF) to PROX and is the sum of the two strongly anti-Robinson matrices
% TARGONE and TARGTWO based on the two row and column
% object orderings given by the ending permutations OUTPERMONE
% and OUTPERMTWO. KBLOCK defines the block size in the use the
% iterative quadratic assignment routine.

function [outpermone,outpermtwo,coordone,coordtwo,fitone,fittwo,addconone,addcontwo,vaf] =
...
biscalqa(prox,targone,targtwo,inpermone,inpermtwo,kblock,nopt)

%BISCALQA carries out a bidimensional scaling of a symmetric proximity
%  matrix using iterative quadratic assignment.
%  PROX is the input proximity matrix (with a zero main diagonal and a
%  dissimilarity interpretation);
%  TARGONE is the input target matrix for the first dimension (usually with
%  a zero main diagonal and with a dissimilarity interpretation representing
%  equally-spaced locations along a continuum); TARGTWO is the input target
%  matrix for the second dimension;
%  INPERMONE is the input beginning permutation for the first dimension
%  (a permuation of the first $n$ integers); INPERMTWO is the input beginning
%  permutation for the second dimension;
%  the insertion and rotation routines use from 1 to KBLOCK
%  (which is less than or equal to $n-1$) consecutive objects in
%  the permutation defining the row and column orders of the data matrix.
%  NOPT controls the confirmatory or exploratory fitting of the unidimensional
%  scales; a value of NOPT = 0 will fit in a confirmatory manner the two scales
%  indicated by INPERMONE and INPERMTWO; a value of NOPT = 1 uses iterative QA
%  to locate the better permutations to fit;
%  OUTPERMONE is the final object permutation for the first dimension;
%  OUTPERMTWO is the final object permutation for the second dimension;
%  COORDONE is the set of first dimension coordinates in ascending order;
%  COORDTWO is the set of second dimension coordinates in ascending order;
%  ADDCONONE is the additive constant for the first dimensional model;
%  ADDCONTWO is the additive constant for the second dimensional model;
%  VAF is the variance-accounted-for in PROX by the bidimensional scaling.

function [find,vaf,targone,targtwo,outpermone,outpermtwo, ...
     rowpermone,colpermone,rowpermtwo,colpermtwo,addconone,...
     addcontwo,coordone,coordtwo,axes] = biscaltmac(proxtm,inpermone,inpermtwo,kblock)

% BISCALTMAC finds and fits the sum of two linear unidimensional scales using iterative
projection to
% a two-mode proximity matrix in the $L_{2}$-norm based on permutations
% identified through the use of iterative quadratic assignment.
% PROXTM is the input two-mode proximity matrix ($nrow \times ncol$)
% and a dissimilarity interpretation);
% FIND is the least-squares optimal matrix (with variance-accounted-for
% of VAF) to PROXTM and is the sum of the two  matrices
% TARGONE and TARGTWO based on the two row and column
% object orderings given by the ending permutations OUTPERMONE
% and OUTPERMTWO, and in turn ROWPERMONE and ROWPERMTWO and COLPERMONE
% and COLPERMTWO. KBLOCK defines the block size in the use the
% iterative quadratic assignment routine and ADDCONONE and ADDCONTWO are
% the two additive constants for the two model components; The $n$ coordinates
% are in COORDONE and COORDTWO.  The input permutations are INPERMONE and
% INPERMTWO.  The $n \times 2$ matrix AXES gives the plotting coordinates for the
% combined row and column object set.

function [find,vaf,targone,targtwo] = biultrafnd(prox,inperm)

% BIULTRAFND finds and fits the sum of two ultrametrics using iterative projection
% heuristically on a symmetric proximity matrix in the $L_{2}$-norm.
% PROX is the input proximity matrix (with a zero main diagonal
% and a dissimilarity interpretation);
% INPERM is a permutation that determines the order in which the
% inequality constraints are considered;
% FIND is the found least-squares matrix (with variance-accounted-for
% of VAF) to PROX and is the sum of the two ultrametric matrices TARGONE and TARGTWO.


function [fit,vaf,lengths] = centfit(prox)

% CENTFIT finds the least-squares fitted centroid metric (FIT) to
% PROX, the input proximity matrix (with a zero main diagonal
% and a dissimilarity interpretation);
% The $n$ values that serve to define the approximating sums,
% $g_{i} + g_{j}$, are given in  the vector LENGTHS of size n x 1.

function [fit,vaf,lengths] = centfittm(proxtm)

% CENTFITTM finds the least-squares fitted two-mode centroid metric (FIT) to
% PROXTM, the two-mode rectangular input proximity matrix (with
% a dissimilarity interpretation);
% The $n$ values (where $n$ = number of rows + number of columns)
% serve to define the approximating sums,
% $u_{i} + v_{j}$, where the $u_{i}$ are for the rows and the $v_{j}$
% are for the columns; these are given in  the vector LENGTHS of size n x 1,
% with row values first followed by the column values.

function [fit, diff] = cirfit(prox,inperm)

%CIRFIT does a confirmatory fitting of a given order
%  (assumed to reflect a circular ordering around a closed
%  unidimensional structure) using Dykstra's
%  (Kaczmarz's) iterative projection least-squares method.
%  INPERM is the given order; FIT is an $n \times n$ matrix that
%  is fitted to PROX(INPERM,INPERM) with least-squares value DIFF.

function [fit, vaf, addcon] = cirfitac(prox,inperm)
```

```
%CIRFITAC does a confirmatory fitting (including
%  the estimation of an additive constant) for a given order
%  (assumed to reflect a circular ordering around a closed
%  unidimensional structure) using Dykstra's
%  (Kaczmarz's) iterative projection least-squares method.
%  INPERM is the given order; FIT is an $n \times n$ matrix that
%  is fitted to PROX(INPERM,INPERM) with variance-accounted-for of
%  VAF; ADDCON is the estimated additive constant.

function [fit, vaf, addcon] = cirfitac_ftarg(prox,inperm,targ)

%CIRFITAC_FTARG does a confirmatory fitting (including
%  the estimation of an additive constant) for a given order
%  (assumed to reflect a circular ordering around a closed
%  unidimensional structure) using Dykstra's
%  (Kaczmarz's) iterative projection least-squares method.
%  The inflection points are implicitly given by TARG which
%  is assumed to reflect a circular ordering of the same size as PROX.
%  INPERM is the given order; FIT is an $n \times n$ matrix that
%  is fitted to PROX(INPERM,INPERM) with variance-accounted-for of
%  VAF; ADDCON is the estimated additive constant.

function [fit, vaf] = cirarobfit(prox,inperm,targ)

% CIRAROBFIT fits a circular anti-Robinson matrix using iterative projection to
% a symmetric proximity matrix in the $L_{2}$-norm.
% PROX is the input proximity matrix ($n \times n$ with a zero main diagonal
% and a dissimilarity interpretation);
% INPERM is a given permutation of the first $n$ integers (around a circle);
% TARG is a given $n \times n$ matrix having the circular anti-Robinson
% form that guides the direction in which distances are taken around the circle.
% FIT is the least-squares optimal matrix (with variance-accounted-for
% of VAF) to PROX having an circular anti-Robinson form for the row and column
% object ordering given by INPERM.

function [fit, vaf] = cirsarobfit(prox,inperm,target)

% CIRSAROBFIT fits a strongly circular anti-Robinson matrix using iterative projection to
% a symmetric proximity matrix in the $L_{2}$-norm.
% PROX is the input proximity matrix ($n \times n$ with a zero main diagonal
% and a dissimilarity interpretation);
% INPERM is a given permutation of the first $n$ integers (around a circle);
% TARGET is a given $n \times n$ matrix having the circular anti-Robinson
% form that guides the direction in which distances are taken around the circle.
% FIT is the least-squares optimal matrix (with variance-accounted-for
% of VAF) to PROX having a strongly circular anti-Robinson form for the row and column
% object ordering given by INPERM.

function [fit, vaf, outperm] = cirarobfnd(prox, inperm, kblock)

% CIRAROBFND fits a circular anti-Robinson matrix using iterative projection to
% a symmetric proximity matrix in the $L_{2}$-norm based on a permutation
% identified through the use of iterative quadratic assignment.
% PROX is the input proximity matrix ($n \times n$ with a zero main diagonal
% and a dissimilarity interpretation);
% INPERM is a given starting permutation (assumed to be around the
% circle) of the first $n$ integers;
% FIT is the least-squares optimal matrix (with variance-accounted-for
% of VAF) to PROX having a circular anti-Robinson form for the row and column
% object ordering given by the ending permutation OUTPERM. KBLOCK
% defines the block size in the use the iterative quadratic assignment
% routine.

function [fit, vaf, outperm] = cirsarobfnd(prox, inperm, kblock)

% CIRSAROBFND fits a strongly circular anti-Robinson matrix using iterative projection to
% a symmetric proximity matrix in the $L_{2}$-norm based on a permutation
% identified through the use of iterative quadratic assignment.
% PROX is the input proximity matrix ($n \times n$ with a zero main diagonal
% and a dissimilarity interpretation);
% INPERM is a given starting permutation (assumed to be around the
% circle) of the first $n$ integers;
% FIT is the least-squares optimal matrix (with variance-accounted-for
% of VAF) to PROX having a strongly circular anti-Robinson form for the row and column
% object ordering given by the ending permutation OUTPERM. KBLOCK
% defines the block size in the use the iterative quadratic assignment
% routine.

function [fit, vaf, outperm] = cirarobfnd_ac(prox, inperm, kblock)

% CIRAROBFND fits a circular anti-Robinson matrix using iterative projection to
% a symmetric proximity matrix in the $L_{2}$-norm based on a permutation
% identified through the use of iterative quadratic assignment.
% PROX is the input proximity matrix ($n \times n$ with a zero main diagonal
% and a dissimilarity interpretation);
% INPERM is a given starting permutation (assumed to be around the
% circle) of the first $n$ integers;
% FIT is the least-squares optimal matrix (with variance-accounted-for
% of VAF) to PROX having a circular anti-Robinson form for the row and column
% object ordering given by the ending permutation OUTPERM. KBLOCK
% defines the block size in the use the iterative quadratic assignment
% routine.

function [fit, vaf, outperm] = cirsarobfnd_ac(prox, inperm, kblock)

% CIRSAROBFND fits a strongly circular anti-Robinson matrix using iterative projection to
% a symmetric proximity matrix in the $L_{2}$-norm based on a permutation
% identified through the use of iterative quadratic assignment.
% PROX is the input proximity matrix ($n \times n$ with a zero main diagonal
% and a dissimilarity interpretation);
% INPERM is a given starting permutation (assumed to be around the
% circle) of the first $n$ integers;
% FIT is the least-squares optimal matrix (with variance-accounted-for
% of VAF) to PROX having a strongly circular anti-Robinson form for the row and column
% object ordering given by the ending permutation OUTPERM. KBLOCK
% defines the block size in the use the iterative quadratic assignment
% routine.

function [outperm, rawindex, allperms, index] = ...
  insertqa(prox, targ, inperm, kblock)
```

```
% INSERTQA carries out an iterative Quadratic Assignment maximization task using the
% insertion of from 1 to KBLOCK (which is less than or equal to $n-1$) consecutive objects in
% the permutation defining the row and column order of the data matrix.
% INPERM is the input beginning permutation (a permuation of the first $n$ integers).
% PROX is the $n \times n$ input proximity matrix.
% TARG is the $n \times n$ input target matrix.
% OUTPERM is the final permutation of PROX with the cross-product index RAWINDEX
% with respect to TARG.
% ALLPERMS is a cell array containing INDEX entries corresponding to all the
% permutations identified in the optimization from ALLPERMS{1} = INPERM to
% ALLPERMS{INDEX} = OUTPERM.

function [fit, diff, coord] = linfit(prox,inperm)

%LINFIT does a confirmatory fitting of a given unidimensional order using Dykstra's
%  (Kaczmarz's) iterative projection least-squares method.
%  INPERM is the given order;
%  FIT is an $n \times n$ matrix that is fitted to PROX(INPERM,INPERM) with
%  least-squares value DIFF;
%  COORD gives the ordered coordinates whose absolute differences
%  could be used to reconstruct FIT.

function [fit, vaf, coord, addcon] = linfitac(prox,inperm)

%LINFITAC does a confirmatory fitting of a given unidimensional order
%  using the Dykstra-Kaczmarz iterative projection least-squares method,
%  but differing from LINFIT.M in including the estimation of an additive
%  constant.
%  INPERM  is the given order;
%  FIT is an $n \times n$ matrix that is fitted to PROX(INPERM,INPERM) with
%  variance-accounted-for VAF;
%  COORD gives the ordered coordinates whose absolute differences
%  could be used to reconstruct FIT; ADDCON is the estimated additive constant
%  that can be interpreted as being added to PROX.

function [fit,diff,rowperm,colperm,coord] = linfittm(proxtm,inperm)

%LINFITTM does a confirmatory two-mode fitting of a given unidimensional ordering
%  of the row and column objects of a two-mode proximity matrix
%  PROXTM using Dykstra's (Kaczmarz's) iterative projection least-squares method.
%  INPERM is the given ordering of the row and column objects together;
%  FIT is an nrow (number of rows) by ncol (number of columns) matrix
%  of absolute coordinate differences that is fitted
%  to PROXTM(ROWPERM,COLPERM) with DIFF being the (least-squares criterion) sum of
%  squared discrepancies between FIT and PROXTM(ROWPERM,COLMEAN);
%  ROWPERM and COLPERM are the row and column object orderings derived
%  from INPERM.  The nrow + ncol coordinates (ordered with the smallest
%  set at a value of zero) are given in COORD.

function [fit,vaf,rowperm,colperm,addcon,coord] = linfittmac(proxtm,inperm)

%LINFITTMAC does a confirmatory two-mode fitting of a given unidimensional ordering
%  of the row and column objects of a two-mode proximity matrix
%  PROXTM using Dykstra's (Kaczmarz's) iterative projection least-squares method;
%  it differs from LINFITTM.M  by including the estimation of an additive constant.
%  INPERM is the given ordering of the row and column objects together;
%  FIT is an nrow (number of rows) by ncol (number of columns) matrix
%  of absolute coordinate differences that is fitted
%  to PROXTM(ROWPERM,COLPERM) with VAF being the variance-accounted-for.
%  ROWPERM and COLPERM are the row and column object orderings derived
%  from INPERM.  ADDCON is the estimated additive constant
%  that can be interpreted as being added to PROXTM (or alternatively subtracted
%  from the fitted matrix FIT).  The nrow + ncol coordinates (ordered with the smallest
%  set at a value of zero) are given in COORD.
```

```
function [outperm,rawindex,allperms,index] =  order(prox,targ,inperm,kblock)
```

```
% ORDER carries out an iterative Quadratic Assignment maximization task using
% a given square ($n x n$) proximity matrix PROX (with a zero main diagonal and
% a dissimilarity interpretation).
% Three separate local operations are used to permute
% the rows and columns of the proximity matrix to maximize the cross-product
% index with respect to a given square target matrix TARG:
% pairwise interchanges of objects in the permutation defining the row and column
% order of the square proximity matrix; the insertion of from 1 to KBLOCK
% (which is less than or equal to $n-1$) consecutive objects in
% the permutation defining the row and column order of the data matrix; the
% rotation of from 2 to KBLOCK (which is less than or equal to $n-1$) consecutive objects in
% the permutation defining the row and column order of the data matrix.
% INPERM is the input beginning permutation (a permuation of the first $n$ integers).
% OUTPERM is the final permutation of PROX with the cross-product index RAWINDEX
% with respect to TARG. ALLPERMS is a cell array containing INDEX
% entries corresponding to all the
% permutations identified in the optimization from ALLPERMS{1} = INPERM to
% ALLPERMS{INDEX} = OUTPERM.
```

```
function [outperm, rawindex, allperms, index, squareprox] = ...
  ordertm(proxtm, targ, inperm, kblock)
```

```
% ORDERTM carries out an iterative Quadratic Assignment maximization task using the
% two-mode proximity matrix PROXTM (with entries deviated from the mean proximity)
% in the upper-right- and lower-left-hand portions of
% a defined square ($n x n$) proximity matrix
% (called SQUAREPROX with a dissimilarity interpretation)
% with zeros placed elsewhere (n = number of rows +
% number of columns of PROXTM = nrow + ncol);
% three separate local operations are used to permute
% the rows and columns of the square proximity matrix to maximize the cross-product
% index with respect to a square target matrix TARG:
% pairwise interchanges of objects in the permutation defining the row and column
% order of the square proximity matrix; the insertion of from 1 to KBLOCK
% (which is less than or equal to $n-1$) consecutive objects in
% the permutation defining the row and column order of the data matrix; the
% rotation of from 2 to KBLOCK (which is less than or equal to $n-1$) consecutive objects in
% the permutation defining the row and column order of the data matrix.
% INPERM is the input beginning permutation (a permuation of the first $n$ integers).
% PROXTM is the two-mode $nrow x ncol$ input proximity matrix.
% TARG is the $n x n$ input target matrix.
% OUTPERM is the final permutation of SQUAREPROX with the cross-product index
RAWINDEX
% with respect to TARG. ALLPERMS is a cell array containing INDEX
% entries corresponding to all the
% permutations identified in the optimization from ALLPERMS{1} = INPERM to
% ALLPERMS{INDEX} = OUTPERM.

function [outperm, rawindex, allperms, index] = ...
   pairwiseqa(prox, targ, inperm)

% PAIRWISEQA carries out an iterative Quadratic Assignment maximization task using the
% pairwise interchanges of objects in the permutation defining the row and column
% order of the data matrix.
% INPERM is the input beginning permutation (a permuation of the first $n$ integers).
% PROX is the $n \times n$ input proximity matrix.
% TARG is the $n \times n$ input target matrix.
% OUTPERM is the final permutation of PROX with the cross-product index RAWINDEX
% with respect to TARG.
% ALLPERMS is a cell array containing INDEX entries corresponding to all the
% permutations identified in the optimization from ALLPERMS{1} = INPERM to
% ALLPERMS{INDEX} = OUTPERM.

function [monproxpermut, vaf, diff] = proxmon(proxpermut, fitted)

%PROXMON produces a monotonically transformed proximity matrix (MONPROXPERMUT)
%  from the order constraints obtained from each pair of entries in the input
%  proximity matrix PROXPERMUT (symmetric with a zero main diagonal and a dissimilarity
%  interpretation).
%  MONPROXPERMUT is close to the $n \times n$ matrix FITTED in the least-squares sense;
%  The variance accounted for (VAF) is how much variance in MONPROXPERMUT can be
accounted for by
%  FITTED; DIFF is the value of the least-squares criterion.

function [monproxpermuttm, vaf, diff] = proxmontm(proxpermuttm, fittedtm)

%PROXMONTM produces a monotonically transformed two-mode proximity matrix
(MONPROXPERMUTTM)
%  from the order constraints obtained from each pair of entries in the input two-mode
%  proximity matrix PROXPERMUTTM (with a dissimilarity interpretation).
%  MONPROXPERMUTTM is close to the $nrow \times ncol$ matrix FITTEDTM in the least-
squares sense;
%  The variance accounted for (VAF) is how much variance in MONPROXPERMUTTM
%  can be accounted for by FITTEDTM; DIFF is the value of the least-squares criterion.

function [randproxtm] = proxrandtm(proxtm)

%PROXRANDTM produces a two-mode proximity matrix having
%  entries that are a random permutation of those in the two-mode input proximity
%  matrix PROXTM.

function [stanproxtm, stanproxmulttm] = proxstdtm(proxtm,mean)

%PROXSTDTM produces a standardized two-mode proximity matrix (STANPROXTM) from the
input
%  $nrow \times ncol$ two-mode proximity matrix (PROXTM) with a dissimilarity
%  interpretation.
%  STANPROXTM entries have unit variance (standard deviation of one) with a
%  mean of MEAN given as an input number;
%  STANPROXMULTTM entries have a sum of squares equal to
%  $nrow*rcol$.
```

```
function [stanprox, stanproxmult] = proxstd(prox,mean)

%PROXSTD produces a standardized proximity matrix (STANPROX) from the input
%  $n \times n$ proximity matrix (PROX) with zero main diagonal and a dissimilarity
%  interpretation.
%  STANPROX entries have unit variance (standard deviation of one) with a
%  mean of MEAN given as an input number;
%  STANPROXMULT (upper-triangular) entries have a sum of squares equal to
%  $n(n-1)/2$.

function [prox, targlin, targcir] = ransymat(n)

%  RANSYMAT produces a random symmetric proximity matrix of size
%  $n \times n$, plus two fixed patterned symmetric proximity
%  matrices, all with zero main diagonals.
%  The size of all the generated matrices is n.
%  PROX is symmetric with a zero main diagonal and entries uniform
%  between 0 and 1.
%  TARGLIN contains distances between equally and unit-spaced positions
%  along a line: targlin(i,j) = abs(i-j).
%  TARGCIR contains distances between equally and unit-spaced positions
%  along a circle: targcir(i,j) = min(abs(i-j),n-abs(i-j)).

function [outperm, rawindex, allperms, index] = ...
   rotateqa (prox, targ, inperm, kblock)

% ROTATEQA carries out an iterative Quadratic Assignment maximization task using the
% rotation of from 2 to KBLOCK (which is less than or equal to $n-1$) consecutive objects in
% the permutation defining the row and column order of the data matrix.
% INPERM is the input beginning permutation (a permuation of the first $n$ integers).
% PROX is the $n \times n$ input proximity matrix.
% TARG is the $n \times n$ input target matrix.
% OUTPERM is the final permutation of PROX with the cross-product index RAWINDEX
% with respect to TARG.
% ALLPERMS is a cell array containing INDEX entries corresponding to all the
% permutations identified in the optimization from ALLPERMS{1} = INPERM to
% ALLPERMS{INDEX} = OUTPERM.

function [fit, vaf] = sarobfit(prox, inperm)

% SAROBFIT fits a strongly anti-Robinson matrix using iterative projection to
% a symmetric proximity matrix in the $L_{2}$-norm.
% PROX is the input proximity matrix ($n \times n$ with a zero main diagonal
% and a dissimilarity interpretation);
% INPERM is a given permutation of the first $n$ integers;
% FIT is the least-squares optimal matrix (with variance-accounted-for
% of VAF) to PROX having a strongly anti-Robinson form for the row and column
% object ordering given by INPERM.

function [fit, vaf, outperm] = sarobfnd(prox, inperm, kblock)

% SAROBFND fits a strongly anti-Robinson matrix using iterative projection to
% a symmetric proximity matrix in the $L_{2}$-norm based on a permutation
% identified through the use of iterative quadratic assignment.
% PROX is the input proximity matrix ($n \times n$ with a zero main diagonal
% and a dissimilarity interpretation);
% INPERM is a given starting permutation of the first $n$ integers;
% FIT is the least-squares optimal matrix (with variance-accounted-for
% of VAF) to PROX having a strongly anti-Robinson form for the row and column
% object ordering given by the ending permutation OUTPERM. KBLOCK
% defines the block size in the use the iterative quadratic assignment
% routine.

function [outpermone,outpermtwo,outpermthree,coordone,coordtwo,coordthree, ...
   fitone,fittwo,fitthree,addconone,addcontwo,addconthree,vaf] = ...
triscalqa(prox,targone,targtwo,targthree,inpermone,inpermtwo,inpermthree,kblock,nopt)


%TRISCALQA carries out a tridimensional scaling of a symmetric proximity
%  matrix using iterative quadratic assignment.
%  PROX is the input proximity matrix (with a zero main diagonal and a
%  dissimilarity interpretation);
%  TARGONE is the input target matrix for the first dimension (usually with
%  a zero main diagonal and with a dissimilarity interpretation representing
%  equally-spaced locations along a continuum); TARGTWO is the input target
%  matrix for the second dimension; TARGTHREE is the input target matrix
%  for the third dimension;
%  INPERMONE is the input beginning permutation for the first dimension
%  (a permuation of the first $n$ integers); INPERMTWO is the input beginning
%  permutation for the second dimension; INPERMTHREE is the input beginning
%  permutation for the third dimension;
%  the insertion and rotation routines use from 1 to KBLOCK
%  (which is less than or equal to $n-1$) consecutive objects in
%  the permutation defining the row and column orders of the data matrix;
%  NOPT controls the confirmatory or exploratory fitting of the unidimensional
%  scales; a value of NOPT = 0 will fit in a confirmatory manner the three scales
%  indicated by INPERMONE and INPERMTWO; a value of NOPT = 1 uses iterative QA
%  to locate the better permutations to fit.
%  OUTPERMONE is the final object permutation for the first dimension;
%  OUTPERMTWO is the final object permutation for the second dimension;
%  OUTPERMTHREE is the final object permutation for the third dimension;
%  COORDONE is the set of first dimension coordinates in ascending order;
%  COORDTWO is the set of second dimension coordinates in ascending order;
%  COORDTHREE is the set of third dimension coordinates in asceding order;
%  ADDCONONE is the additive constant for the first dimensional model;
%  ADDCONTWO is the additive constant for the second dimensional model;
%  ADDCONTHREE is the additive constant for the third dimensional model;
%  VAF is the variance-accounted-for in PROX by the bidimensional scaling.

function [outpermone,outpermtwo,outpermthree,coordone,coordtwo,coordthree, ...
   fitone,fittwo,fitthree,addconone,addcontwo,addconthree,vaf,monprox] = ...
trimonscalqa(prox,targone,targtwo,targthree,inpermone,inpermtwo, ...
inpermthree,kblock,nopt)

%TRIMONSCALQA carries out a tridimensional scaling of a symmetric proximity
%  matrix using iterative quadratic assignment, plus it provides an
%  optimal monotonic transformation (MONPROX) of the original input
%  proximity matrix.
%  PROX is the input proximity matrix (with a zero main diagonal and a
%  dissimilarity interpretation);
%  TARGONE is the input target matrix for the first dimension (usually with
%  a zero main diagonal and with a dissimilarity interpretation representing
%  equally-spaced locations along a continuum); TARGTWO is the input target
%  matrix for the second dimension; TARGTHREE is the input target matrix
%  for the third dimension;
%  INPERMONE is the input beginning permutation for the first dimension
%  (a permuation of the first $n$ integers); INPERMTWO is the input beginning
%  permutation for the second dimension; INPERMTHREE is the input
%  beginning permutation for the third dimension;
%  the insertion and rotation routines use from 1 to KBLOCK
%  (which is less than or equal to $n-1$) consecutive objects in
%  the permutation defining the row and column orders of the data matrix;
%  NOPT controls the confirmatory or exploratory fitting of the unidimensional
%  scales; a value of NOPT = 0 will fit in a confirmatory manner the two scales
%  indicated by INPERMONE and INPERMTWO; a value of NOPT = 1 uses iterative QA
%  to locate the better permutations to fit;
%  OUTPERMONE is the final object permutation for the first dimension;
%  OUTPERMTWO is the final object permutation for the second dimension;
%  OUTPERMTHREE is the final object permutation for the third dimension;
%  COORDONE is the set of first dimension coordinates in ascending order;
%  COORDTWO is the set of second dimension coordinates in ascending order;
%  COORDTHREE is the set of second dimension coordinates in ascending order;
%  ADDCONONE is the additive constant for the first dimensional model;
%  ADDCONTWO is the additive constant for the second dimensional model;
%  ADDCONTHREE is the additive constant for the second dimensional model;
%  VAF is the variance-accounted-for in MONPROX by the tridimensional scaling.

function [targlin] = targlin(n)

%  TARGLIN produces a symmetric proximity matrix of size
%  $n \times n$, containing distances between equally and unit-spaced positions
%  along a line: targlin(i,j) = abs(i-j).

function [targcir] = targcir(n)

%  TARGCIR produces a symmetric proximity matrix of size
%  $n \times n$, containing distances between equally and unit-spaced positions
%  along a circle: targcir(i,j) = min(abs(i-j),n-abs(i-j)).

function [fit, vaf] = targfit(prox,targ)

% TARGFIT fits through iterative projection a given set of equality and
% inequality constraints (as represented by the equalities and
% inequalities present among the entries in a target matrix
% TARG) to a symmetric proximity matrix in the $L_{2}$-norm.
% PROX is the input proximity matrix (with a zero main diagonal
% and a dissimilarity interpretation);
% TARG is a matrix of the same size as PROX;
% FIT is the least-squares optimal matrix (with variance-accounted-for
% of VAF) to PROX satisfying the equality and
% inequality constraints implicit in TARG.

function [ultracomp] = ultracomptm(ultraproxtm)

% ULTRACOMPTM provides a completion of a given two-mode ultrametric matrix
% to a symmetric proximity matrix satisfying the usual ultrametric
% constraints.
% ULTRAPROXTM is the $nrow \times ncol$ two-mode ultrametric matrix;
% ULTRACOMP is the completed symmetric $n \times n$ proximity matrix having the usual
% ultrametric pattern, for $n = nrow + ncol$.

function [fit,vaf] = ultrafit(prox,targ)

% ULTRAFIT fits a given ultrametric using iterative projection to
% a symmetric proximity matrix in the $L_{2}$-norm.
% PROX is the input proximity matrix (with a zero main diagonal
% and a dissimilarity interpretation);
% TARG is an ultrametric matrix of the same size as PROX;
% FIT is the least-squares optimal matrix (with variance-accounted-for
% of VAF) to PROX satisfying the ultrametric constraints implicit in TARG.

function [fit,vaf] = ultrafittm(proxtm,targ)

% ULTRAFITTM fits a given (two-mode) ultrametric using iterative projection to
% a two-mode (rectangular) proximity matrix in the $L_{2}$-norm.
% PROXTM is the input proximity matrix (with a dissimilarity interpretation);
% TARG is an ultrametric matrix of the same size as PROXTM;
% FIT is the least-squares optimal matrix (with variance-accounted-for
% of VAF) to PROXTM satisfying the ultrametric constraints implicit in TARG.

function [find,vaf] = ultrafnd(prox,inperm)

% ULTRAFND finds and fits an ultrametric using iterative projection
% heuristically on a symmetric proximity matrix in the $L_{2}$-norm.
% PROX is the input proximity matrix (with a zero main diagonal
% and a dissimilarity interpretation);
% INPERM is a permutation that determines the order in which the
% inequality constraints are considered;
% FIND is the found least-squares matrix (with variance-accounted-for
% of VAF) to PROX satisfying the ultrametric constraints.

function [find,vaf] = ultrafndtm(proxtm,inpermrow,inpermcol)

% ULTRAFNDTM finds and fits a two-mode ultrametric using iterative projection
% heuristically on a rectangular proximity matrix in the $L_{2}$-norm.
% PROXTM is the input proximity matrix (with a dissimilarity interpretation);
% INPERMROW and INPERMCOL are permutations for the row and column
% objects that determine the order in which the
% inequality constraints are considered;
% FIND is the found least-squares matrix (with variance-accounted-for
% of VAF) to PROXTM satisfying the ultrametric constraints.

function [orderprox,orderperm] = ultraorder(prox)

% ULTRAORDER finds for the input proximity matrix PROX
% (assumed to be ultrametric with a zero main diagonal),
% a permutation ORDERPERM that displays the anti-
% Robinson form in the reordered proximity matrix
% ORDERPROX; thus, prox(orderperm,orderperm) = orderprox.

function [] = ultraplot(ultra)

%ULTRAPLOT gives a dendrogram plot for the input ultrametric dissimilarity
```

```
%matrix ULTRA.

function [fit, vaf, outperm, addcon] = unicirac(prox, inperm, kblock)

% UNICIRAC finds and fits a circular unidimensional scale using iterative projection to
% a symmetric proximity matrix in the $L_{2}$-norm based on a permutation
% identified through the use of iterative quadratic assignment.
% PROX is the input proximity matrix ($n \times n$ with a zero main diagonal
% and a dissimilarity interpretation);
% INPERM is a given starting permutation (assumed to be around the
% circle) of the first $n$ integers;
% FIT is the least-squares optimal matrix (with variance-accounted-for
% of VAF) to PROX having a circular anti-Robinson form for the row and column
% object ordering given by the ending permutation OUTPERM.  The spacings
% among the objects are given by the diagonal entries in FIT (and
% the extreme (1,n) entry in FIT). KBLOCK
% defines the block size in the use the iterative quadratic assignment
% routine.  The additive constant for the model is given by ADDCON.

function [outperm, rawindex, allperms, index, coord, diff] = ...
   uniscalqa(prox, targ, inperm, kblock)

%UNISCALQA carries out a unidimensional scaling of a symmetric proximity
%  matrix using iterative quadratic assignment.
%  PROX is the input proximity matrix (with a zero main diagonal and a
%  dissimilarity interpretation);
%  TARG is the input target matrix (usually with a zero main diagonal and
%  with a dissimilarity interpretation representing equally-spaced locations
%  along a continuum);
%  INPERM is the input beginning permutation (a permuation of the first $n$ integers).
%  OUTPERM is the final permutation of PROX with the cross-product index RAWINDEX
%  with respect to TARG redefined as $ = \{abs(coord(i) - coord(j))\}$;
%  ALLPERMS is a cell array containing INDEX entries corresponding to all the
%  permutations identified in the optimization from ALLPERMS{1} = INPERM to
%  ALLPERMS{INDEX} = OUTPERM.
%  The insertion and rotation routines use from 1 to KBLOCK
%  (which is less than or equal to $n-1$) consecutive objects in
%  the permutation defining the row and column order of the data matrix.
%  COORD is the set of coordinates of the unidimensional scaling
%  in ascending order;
%  DIFF is the value of the least-squares loss function for the
%  coordinates and object permutation.

function [fit, vaf, outperm, rowperm, colperm, addcon, coord] = uniscaltmac(proxtm, inperm,
kblock)

% UNISCALTMAC finds and fits a linear unidimensional scale using iterative projection to
% a two-mode proximity matrix in the $L_{2}$-norm based on a permutation
% identified through the use of iterative quadratic assignment.
% PROXTM is the input two-mode proximity matrix ($n_{a} \times n_{b}$ with a zero main
diagonal
% and a dissimilarity interpretation);
% INPERM is a given starting permutation of the first $n = n_{a} + n_{b}$ integers;
% FIT is the least-squares optimal matrix (with variance-accounted-for
% of VAF) to PROXTM having a linear unidimensional form for the row and column
% object ordering given by the ending permutation OUTPERM.  The spacings
% among the objects are given by the entries in FIT. KBLOCK
% defines the block size in the use the iterative quadratic assignment
% routine.  The additive constant for the model is given by ADDCON.
% ROWPERM and COLPERM are the resulting row and column permutations for
% the objects.  The nrow + ncol coordinates (ordered with the smallest
% set at a value of zero) are given in COORD.
```