SNPCA DOCUMENTATION

1. Running SNPCA with a driver file

The easiest way to run the SNPCA algorithm is to use a driver file, whose main sections are:

(1) Set algorithm parameters in structure named `SNPCA_params` structure.
(2) Generate the set of surface data point coordinates.
(3) Call `SNPCA_interleaved_main` with surface data point coordinates and SNPCA algorithm parameters

The user can reverse or interleave items 1 and 2. The fields of `SNPCA_params` are described in Section 3.

Here's the body of an SNPCA driver for torus data in $\mathbb{R}^3$ with no noise, with the three main sections separated by comments.

```
function [] = SNPCA_interleaved_torus_local_data_Q()

state_space_dmnsn = 3;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Set SNPCA algorithm parameters in SNPCA_params structure
SNPCA_params = new_SNPCA_params();

SNPCA_params.chrctrstc_lngth                 = .5;
SNPCA_params.cnstrnt_rad_fac                 = .5*sqrt(3);
SNPCA_params.new_tri_max_edg_lngth           = 1.5*SNPCA_params.chrctrstc_lngth;
SNPCA_params.srch_rad_fac1                   = 1.0;
SNPCA_params.srch_rad_fac2                   = 1.0;
SNPCA_params.intl_pt_ind                     = 1;
SNPCA_params.prfrd_cnstrnt_rds               = .5*sqrt(3)*SNPCA_params.chrctrstc_lngth;
SNPCA_params.prfrd_cnstrnt_rds_wght          = .1;
SNPCA_params.non_adj_tri_dist_tol            = .5*SNPCA_params.chrctrstc_lngth;
SNPCA_params.cand_vert_max_nudge_dist        = SNPCA_params.chrctrstc_lngth;
SNPCA_params.adj_vert_max_nudge_dist         = .25*SNPCA_params.chrctrstc_lngth;
SNPCA_params.emprcl_drctn_crrltn_eval_bias = 1e-6;
SNPCA_params.plot_frqncy                     = 1;
SNPCA_params.rtn_mtrx                        = eye(3);
SNPCA_params.nnz_egnvals                     = 3;
SNPCA_params.emprcl_drctn_egn_sprs_algrthm = false;
SNPCA_params.max_num_restarts                = 1;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Generate set of surface data point coordinates
noise_magnitude = 0;
seed = 0;
rnd_strm = RandStream('mt19937ar','seed', seed);
RandStream.setGlobalStream(rnd_strm);
n   = 1e4; %number of data
rh  = 4; %horizontal torus radius
rd  = 1; %vertical torus radius
trs_crdnts = gen_torus_data_pts2(rh, rd, n);

srfc_crdnts = SNPCA_params.rtn_mtrx(:,1:3)*trs_crdnts;

disp(['Torus large radius: ' num2str(rh)])
disp(['Torus small radius: ' num2str(rd)])
disp(['state space dimension: ' num2str(state_space_dmnsn)])

disp(['Number surface data points: ' num2str(n)])
disp(['Noise magnitude: ' num2str(noise_magnitude)])

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Call SNPCA_interleaved_main with surface data point coordinates
%and SNPCA algorithm parameters
[vrtx_crdnts, tri_vrtx_inds, edg_vrtx_inds, ...
    P_dmnt_egnvctrs, P_dmnt_egnvals] ...
    = ...
    SNPCA_interleaved_main(srfc_crdnts, SNPCA_params);
```

2. List of driver files

- SNPCA_cone
- SNPCA_Ford_LIDAR
- SNPCA_sphere
- SNPCA_swiss_roll
- SNPCA_torus
- SNPCA_twisted_sheet

3. Fields of SNPCA_params structure

The function `new_SNPCA_params()` returns an instance of the `SNPCA_params` struct.
`adj_vert_max_nudge_dist`
Tolerance used to determine whether to merge a front edge of the candidate triangle that

is nearly parallel to an adjacent front edge. Typically set to .25 times the characteristic length. The criteria should really be based on the angle of the two edges as well, but is not implemented this way.

`cand_vert_max_nudge_dist`
The maximum allowable distance from the minimizer of the constrained minimization problem to the nearest surface data point. The algorithm rejects the candidate triangle if this distance exceeds the tolerance specified by this parameter. Contrast to `new_tri_max_edg_lngth` in the seam sewing stage.

`chrctrstc_lngth`
The characteristic length scale.

`cnstrnt_rad_fac`
The radius of the constraint sphere expressed as a multiple of the active edge length. Setting this to $\sqrt{3}/2$ allows for an equilateral triangle. The radius of the constraint sphere is calculated as the weighted average of this dynamically computed radius, and a user supplied preferred value of the radius. See entries for `prfrd_cnstrnt_rds` and `prfrd_cnstrnt_rds_wght` to see how the radius is calculated in the code.

`emprcl_drctn_crrltn_eval_bias`
The value of μ in $(P + \mu I)^{-1}$, where P is the empirical direction covariance matrix.

`emprcl_drctn_egn_sprs_algrthm`
Set to `true` if you want Matlab to use a sparse matrix algorithm to calculate the eigen-decomposition of the empirical direction covariance matrix. See entry for `nnz_egnvals`.

`INTL_EDG_FIFO_LNGTH`
The initial amount of space allocated for the edge stack, as measured as the number of edges the edge stack can hold. The edge stack can grow to hold more edges if the number of edges on the stack grows larger than `INTL_EDG_FIFO_LNGTH`. The default set in `new_SNPCA_params()` is 1024.

`intl_pt_ind`
Specifies which surface data point the algorithm should start at. The column indexed by `intl_pt_ind` of the surface data point coordinates matrix contains the coordinates of one of the vertices of the edge shared by the two initial triangles.

`max_num_restarts`
The maximum number of times the algorithm will go through the advancing front and seam sewing stages.

`MAX_NUM_VRTCS`

The maximum number of vertices allowed in the triangulation. The default set in `new_SNPCA_params()` is 2048.

`new_tri_max_edg_lngth`
The maximum edge length of a new triangle created during the seam sewing stage. Contrast to `cand_vert_max_nudge_dist` in the advancing front stage.

`nnz_egnvals`
The number eigenvalues of the empirical direction covariance matrix that are not set to zero inside the code. The sparse eigen-decomposition algorithm (`emprcl_drctn_egn_sprs_algrthm` set to `true`) requires this parameter, and the non-sparse algorithm respects it.

`non_adj_tri_dist_tol`
Inter-triangle distance tolerance used to accept/reject candidate triangle. If the candidate triangle overlaps with an existing triangle and the distance between the two triangles is less than `non_adj_tri_dist_tol`, then the algorithm rejects the candidate triangle.

`path_saved_data`
A string containing the path where the algorithm should save its state after completing an advancing front or seam sewing stage. The default set in `new_SNPCA_params()` is `interleaved_data` followed by the system path separator returned by `filesep()`.

`plot_frqncy`
Set to 0 to suppress plotting while the algorithm is running, otherwise set to a positive number. The "frqncy" in the name is a misnomer. The code treats this variable as a boolean, where values less than or equal to zero correspond to false, and positive values correspond to true.

`prfrd_cnstrnt_rds`
The "preferred" radius of the constraint sphere. Typically set to $\sqrt{3}/2$ times the characteristic length. Unlike the relative radius specified by `cnstrnt_rad_fac`, this value is constant. See entries for `prfrd_cnstrnt_rds_wght` and `cnstrnt_rad_fac`.

`prfrd_cnstrnt_rds_wght`
The weight put on `prfrd_cnstrnt_rds` when calculating the constraint sphere radius as a weighted average of the preferred and nominal values. Set this parameter to zero to make the constraint sphere radius equal to `cnstrnt_rad_fac` times the active edge length.

`rtn_mtrx`
A user specified orthonormal matrix used for embedding points in a subspace of a higher dimensional space, and for plotting points in a higher dimensional space. For example, `rtn_mtrx(:,1:3)*trs_crdnts` embeds the torus surface data points in a higher dimensional space. The algorithm plots `rtn_mtrx(:,1:3).'*vrtx_crdnts` (which is in $\mathbb{R}^3$) when

plotting the triangulation.

`save_data`
A boolean that determines if algorithm should save its state after completing an advancing front and seam sewing stage. The algorithm state data is saved in the directory specified by `path_saved_data`. Set this variable to `true` to ease your debugging efforts.

`srch_rad_fac1`, `srch_rad_fac2`
The radius of the search sphere as a multiple of the active edge length. The empirical direction covariance matrix associated with an active edge vertex is constructed from surface data points inside the search sphere centered at that vertex. Typically set to 1. The reason for having two different parameters is no longer valid. This should be fixed, but does no harm.

4. Important functions

SNPCA_interleaved_main.
Inputs:

- `srfc_crdnts`: Matrix of surface data point coordinates, where the i^{th} column holds the coordinates of the i^{th} surface data point.
- `SNPCA_params`: See section 3.

Outputs:

- `vrtx_crdnts`: Coordinates of the vertices in the triangulation.
- `tri_vrtx_inds`: Three column matrix, where the i^{th} row holds the vertex indices of the i^{th} triangle in the triangulation.
- `edg_vrtx_inds`: Three column matrix, where the i^{th} row holds the vertex indices of the i^{th} edge in the triangulation. The first two entries in a row index the vertices that define the edge. The third entry indexes the interior vertex of the edge, which is not unique if the edge belongs to two triangles. You shouldn't use the third entry (the purpose it served is no longer relevant).
- `P_dmnt_egnvctrs`: Cell array of eigenvector matrices. `P_dmnt_egnvctrs{i}`: holds the eigenvectors of the empirical direction covariance matrix centered at the i^{th} vertex in the triangulation.
- `P_dmnt_egnvals`: Cell array of eigenvalues, stored as column vectors. `P_dmnt_egnvals{i}` holds the eigenvalues of the empirical direction covariance matrix centered at the i^{th} vertex in the triangulation.

advancing_front_main_loop.
Inputs:

- `srfc_crdnts`: See above.
- `edg_fifo`: First-in-first-out stack of edges. Inside the body of `advancing_front_main_loop`, edges are popped off the stack until a front edge is encountered, which becomes the active front edge. If the candidate triangle generated from the active front edge is

accepted, then any new edges are pushed onto the top of the edge stack. The stack is implemented as an array with a pointer to the head (`edg_fifo_ind`).
- `edg_fifo_ind`: Index of the head of `edg_fifo`, i.e. index into the array `edg_fifo` of the next edge to be popped off the stack.
- `vrtx_crdnts`: See above.
- `tri_vrtx_inds`: See above.
- `edg_vrtx_inds`: See above.
- `srfc_pt_ind_to_vrtx_ind`: A map from surface data point indices to vertex indices. Used an associative array since there are typically far more surface data points than vertex indices. If the i^{th} vertex coincides with the j^{th} surface data point, then `i == srfc_pt_ind_to_vrtx_ind(j)` is true.
- `vrtx_ind_to_srfc_pt_ind` A map from vertex indices to surface data point indices. If the i^{th} vertex coincides with the j^{th} surface data point, then `j == vrtx_ind_to_srfc_pt_ind(i)` is true.
- `intl_mnmzr_crdnts`: Cell array of coordinate vectors that solve the constrained minimization problem. `intl_mnmzr_crdnts{i}` holds the coordinates of the minimizer generated at the i^{th} edge.
- `P_dmnt_egnvctrs`: See above.
- `P_dmnt_egnvals`: See above.
- `SNPCA_params`: See Section 3.
- `plot_hndls`: Structure of plot handles with fields
  - `surf_pts_hndl`
  - `tris_hndl`
  - `front_hndls`
  - `actv_edg_hndl`
  - `actv_edg_intr_hndl`
  - `cand_vert_and_surf_hdnl`

Outputs:
- `vrtx_crdnts`
- `tri_vrtx_inds`
- `edg_vrtx_inds`
- `srfc_pt_ind_to_vrtx_ind`
- `vrtx_ind_to_srfc_pt_ind`
- `intl_mnmzr_crdnts`
- `P_dmnt_egnvctrs`
- `P_dmnt_egnvals`

sew_seams_decomp3.
Inputs
- `vrtx_crdnts`
- `tri_vrtx_inds`
- `edg_vrtx_inds`
- `invbl_edg_inds`

- `P_dmnt_egnvctrs`
- `P_dmnt_egnvals`
- `rtn_mtrx`
- `new_tri_max_edg_lngth`
- `non_adj_tri_dist_tol`
- `emprcl_drctn_crrltn_eval_bias`
- `plot_hndls`
- `plot_frqncy`

Outputs

- `tri_vrtx_inds`
- `edg_vrtx_inds`

5. High level advancing front stage

Advancing front stage pseudo code:

```
• While edge stack is non-empty
    (1) Pop edges off edge stack until front edge encountered
    (2) Compute empirical direction covariance matrices associated with each vertex
        of the active front edge
    (3) Compute minimizer of constrained minimization problem
    (4) Nudge minimizer to nearest (in induced metric) surface data point
    (5) Determine if candidate triangle conflicts with existing triangulation
    (6) Push new edges onto edge stack
```

Pop edges (line 1)

- No function, this is done inside a while loop.

Empirical direction covariance matrices (line 2)

- `in_nghbrhd_crdnts`: Get list of points in neighborhood of active edge vertex.
- `pnts_to_egn_dcmp`: Compute eigen-decompositions of the empirical direction covariance matrices associated with each vertex of the active front edge.

Minimizer (line 3)

- `dstnc_objctv_decomp2`: General induced metric, use to build induced metrics associated with active edge vertices that are passed to `fmincon`.
- `cnstrnt_x0`: Objective function for `fmincon`. Needed to get a starting point on the constraint sphere that satisfies the isosceles constraint.
- `gen_tri_vert_coords3_decomp`: Wrapper for `fmincon` that generates the minimizer to the constrained minimization problem.

Nudge (line 4)

- `nearest_srfc_pt_Q2`: Finds the nearest surface data point in the metric induced by the data near the minimizer.

Conflict (line 5)

- `cndt_adjcnt_vrtx_indxs`: Two edges are adjacent if they share a vertex. This function returns a list of vertices that belong to edges adjacent to the active front edge, but do not belong to the active front edge.
- `nrby_frnt_edgs`: Returns a list of edge indices that are near the active front edge.
- `cand_vert_error_tmp`: Determines if the candidate triangle conflicts with a triangle in the existing triangulation.

6. High level seam sewing stage

The set of front edges and their vertices may be viewed as a graph, and the seam sewing stage algorithm decomposes this graph into a set of simple cycles for the purpose of determining the order in which the front edges should be visited. The seam sewing stage initializes the edge stack by pushing every edge from each simple cycle onto the stack, where edges that are contiguous in the stack are adjacent in the front. It is not essential to the algorithm to decompose the front into simple cycles, but doing so tends to close holes in the triangulation quickly.

The seam sewing stage terminates when there are no front edges (ideally), or when it is impossible to generated a non-conflicting triangle on any front edge.

sew_seams_decomp4. All the input and output arguments are defined elsewhere in this document.
Inputs:

- `vrtx_crdnts`
- `tri_vrtx_inds`
- `edg_vrtx_inds`
- `invbl_edg_inds`
- `P_dmnt_egnvctrs`
- `P_dmnt_egnvals`
- `new_tri_max_edg_lngth`
- `non_adj_tri_dist_tol`
- `emprcl_drctn_crrltn_eval_bias`
- `rtn_mtrx`
- `plot_hndls`
- `plot_frqncy`

Outputs:

- `tri_vrtx_inds`
- `edg_vrtx_inds`

The two main functions called by `sew_seams_decomp3` are:

- `frnt_cycl_basis`: Decomposes the front into a set of simple cycles.
- `new_tris_frm_frnt_edg`: Generates new triangles from front edges and existing vertices.

7. Known issues and bugs

Bug: The initial triangles generated after the first advancing front/seam sewing stage can conflict without being rejected. This can result in two triangles fitting the same subset of data points.

Issue: The definition of two triangles conflicting is not symmetric, i.e. triangle T_1 can conflict with triangle T_2, but triangle T_2 does not conflict with triangle T_1.

Issue: Two nearby triangles that have no edges in common can be nonconflicting, but the algorithm fails to close the seam between them. I think this can be remedied by changing the definition of conflict to mark two triangles with no shared vertices as conflicting if their distance is less than some tolerance.