ATLAS IMPLEMENTATION: USER GUIDE

1. Code Summary

`construction(S,init,delta,rho,m,p,t_0,d)`

- inputs
  - `S` - SDE simulator, see `simulator.m` for example
  - `delta` - homogenization scale, affects density of sample net
  - `rho` - given distance function
  - `m` - desired number of landmarks to generate for each net point
  - `p` - num sample paths for each point in net
  - `t_0` - desired time of simulation for each call to `S`
  - `d`- intrinsic dimension of dynamics
  - `net_info` - struct for delta-net, contains
    - `net` - output $\delta-$net, columns are data points,
    - `deg` - $N \times 1$ vector, entry j is the number of neighbors (degree) of net point j
    - `max_deg` - maximum entry of `deg`
    - `neighbors` - $N \times max_deg$ array, row j is neighbor indices of net point j, with 0's after $deg(j)$ index
- outputs:
  - `new_Sim` - struct with key values that specify parameters for simulating a constant coefficient SDE

Constructs the landmarks, ATLAS charts, and local SDE simulators for a given delta net and simulator. Can be run with input parameters, or with no parameters and it will load from `current_driver.mat`.

The output is a struct `new_S` containing parameters for the family of local SDEs learned by the ATLAS algorithm. In particular, each net point n has a corresponding local simulator for the constant coefficient SDE

$$dX_t = b_n dt + \sigma_n dW$$

, and any neighboring net points $j \sim k$ have associated transition maps T_{jk}, T_{kj} which allow the global ATLAS simulator to switch between the two local SDE simulators.

The struct `new_S` has key values:

- `c` - $d \times N \times N$ array containing local coordinates of neighboring charts, updated each iteration of `construct_local_SDE`, `c(:,i,j)` is a $d \times 1$ vector, is j-th neighbor of net point n, expressed in chart n coordinates
- `b` - $d \times 1$ vector, drift coordinate for n's local SDE
- `sigma` - $d \times d \times N$ array, `sigma(:,:,n)` is $d \times d$ matrix of diffusion coords for n's local SDE, `mu` - $d \times N \times N$ array containing mean landmarks of local neighboring charts `mu(:,i,j)` is a $d \times 1$ vector, the average of chart j's landmark expressed in chart n coords

`learned_simulator_step`

- inputs:
  - `x` - initial point for the simulator, vector in $\mathbb{R}^d$
  - `i` - chart index for `x`
  - `new_S` - new simulator struct, struct containing computed `T,B,C,mu,Phi` structs mentioned above as `new_S.T,,new_S.B` etc.
  - `neighbors` - as above
  - `d` - intrinsic dimension of dynamics
  - `dt` - desired time-step length
  - `delta` - as above
- outputs:
  - `x` - new x value after timestep, vector in $\mathbb{R}^d$
  - `j` - chart index of `x` above, used for future timesteps

`delta_net(net,init,delta,rho,is_random)`

- inputs:
  - `init` - $D \times N$ matrix, set of N vectors in R^d
  - `delta` - coarseness of delta-net
  - `rho` - given distance function
- outputs:
  - `net` - output $\delta-$net, columns are data points,
  - `deg` - $N \times 1$ vector, entry j is the number of neighbors (degree) of net point j
  - `max_deg` - maximum entry of `deg`
  - `neighbors` - $N \times maxdeg$ array, row j is neighbor indices of net point j, with 0's after $deg(j)$ index

This function takes a set of points `init` and :

- Sub-samples the given set of points to create a delta-net, i.e. a set of points Γ in a domain $\mathcal{M}$ such that
  - $d(y_i, y_j) > \delta$ for all $y_i, y_j \in \Gamma$
  - Given $x \in \mathcal{M}$, there exists $y \in \Gamma$ with $d(x,y) < \delta$.
- identifies neighboring points in the delta net, i.e net points with $d(y_i, y_j) < 2(\delta)$.

`LMDS(L,Z,rho,d)`

- inputs:
  - `L` - set of landmarks for Z, array with columns in $\mathbb{R}^D$ as landmark vectors
  - `Z` - set of data points, array with columns in $\mathbb{R}^D$ as data vectors
  - `rho` - given distance function
  - `d` - intrinsic dimension, LMDS projects columns of Z, L onto R^d
- outputs:
  - `embed_L` - MDS output for landmarks, array with columns in $\mathbb{R}^d$ as projected landmark vectors
  - `embed_Z` - projected Z data, array with columns in $\mathbb{R}^d$ as projected data vectors

Implementation of the Landmark Multi-Dimensional Scaling Algorithm (cite!!) for a given set of landmarks and data

References