This Matlab toolbox reproduces the numerical illustrations of the paper:

> Quantum Optimal Transport for Tensor Field Processing, 2016

Structure of the toolbox

- `test_*.m` : the main scripts to reproduce the figures of the article.
- `test_simple_*.m` : simple tests (display, etc), not related to optimal transport.
- `quantum_*.m` : main computational functions (see bellow).
- `toolbox/` : useful generic helper functions.
- `toolbox_anisotropic/` : high performance resolution of anisotropic diffusions on regular grids (code of Jean-Marie Mirebeau)
- `toolbox_connections/` : computation of smooth vector fields on surface (Keenan Crane)
- `toolbox_fast_marching/` : Fast-Marching for resolution of anisotropic geodesic distances (code of Fethallah Benmansour).

Storage Conventions

Tensor fields are stored in format (d,d,...) for tensors of size (d,d) so the first dimensions exposes the tensors. This is opposed to the convention used for e.g. vizualization helpers tools.

The computational functions (`quantum_sinkhorn` , `quantum_interp` , `quantum_sinkhorn`) takes as input tensors as array of size (d,d,N) where N is the number of grid points. Function specific to 2-D (e.g. display) rather uses the storage format (d,d,n,n) with N=n*n.

Parameters and Accelerations

The two main parameters are:

- `epsilon`: controls the entropic regularization. Increasing epsilon leads to an extra blurring, but a significant acceleration of the convergence. So if the algorithm is running too slow, you should try increasing this value and reducing the number of iterations (`options.niter`).
- `rho`: controls the relaxation of the marginals. If you expect to have data with lots of local or global mass variation (variation of the trace) you should increase it. Note that the interpolation method implemented in `quantum_interp` actually takes into account this mass variation and cancels it, so that at the end of the day, the interpolation is exact (meaning that there is mass creation/destruction during the interpolation, which is what most people would expect it to do).

Main Computational Functions

The main functions are:

- `quantum_sinkhorn`: compute an optimal coupling between two tensor fields.
- `quantum_interp`: use such a coupling to compute an interpolation between the tensor fields.
- `quantum_interp_free`: same but for measures on unstructured point clouds.
- `quantum_barycenter`: compute a barycenter between several fields, extends quantum_sinkhorn.m to several input measures.

Scripts

The main scripts to reproduce the figures of the article are:

- `test_2x2_1d.m` : 2x2 tensors in 1-D (along a line).
- `test_3x3_1d.m` : 3x3 tensors in 1-D (along a line).
- `test_2x2_2d.m` : 2x2 tensors in 2-D (image), also include an anisotropic diffusion display to mimick procedural noise generation.
- `test_2x2_2d_barycenters.m` : same but for barycenters instead of just 2 inputs.
- `test_2x2_meshes.m` : 2x2 tensors on the tangent plane of a mesh.
- `test_2x2_meshes_barycenters.m` : same but for barycenters instead of just 2 inputs.
- `test_dti.m` : application to diffusion tensor imaging.
- `test_meshing.m` : application to triangular mesh generation.
- `test_sparse.m` : sparse solver (see bellow)
- `test_spectral_densities.m` : application to Gaussian texture synthesis.
- `test_simple_*.m` : a bunch of simple tests (display, diffusion, synthesis, etc) not related to optimal transport.

Sparse Solver

The gamma coupling is stored explicitly as a full matrix. To scale to much larger computational setting, it is mandatory to use sparse matrices. This is active if you provide a sparse cost matrix `c` . This means that you have to "guess" beforehand roughly the location of the transport map. This is usually done by running the code in a coarse to fine way.

An test of a sparse multiscale solver is implemented in `test_2x2_2d_sparse.m`. It means that the gamma is first computed on a low resolution grid, and that this low resolution is used to locate the support of a sparse coupling at a higher resolution, and this process it repeated. For "geometric" configuration (i.e. when the input measures are smooth and the cost is also smooth, so that ultimately the coupling should concentrate on a Monge-map), this strategy has proved to be surprisingly effective.

Helpers

Generic helper functions are in `toolbox/`.

Helpers functions for mesh processing are in `toolbox_geometry/`.

Helper functions for the quantum-OT are implemented in `toolbox_quantum/`, in particular:

- `load_helpers`: load a structure filled with many simple functions.
- `LSE`: log-sum-exp operator, at the core of Sinkhorn iterations.
- `expM` / `logM`: fast matrix exp and log.
- `plot_tensors_1d` / `plot_tensors2d`: vizualization
- `load_tensors_pair`: load synthetic 1D and 2D examples.
- `tensor_*`: operations on tensor (multiplication, transposition, identity, etc.)
- `tensor_eigendecomp` / `tensor_eigenrecomp`: diagonalization of tensors.

Mex

You might want to re-compile for your architecture the mex files in `toolbox_quantum/tensor_logexp/`, which are fast C implementation of `logM` / `expM`. The script `compile_mex.m` should be able to make the compilation.

To enable the use of these fast mex file, set

> global logexp_fast_mode logexp_fast_mode = 4; % fast mex

Thanks

- For mesh computations, it uses the Matlab toolbox toolbox_connections/ of Keenan Crane in order to pre-compute a smoothly varying (outside singular points) vector field.
- For the resolution of anisotropic diffision, it uses the anisotropic Laplacian discretization of Jean-Marie Mirebeau).

Copyright