Examples of how to solve semi-discrete optimal transport problems with MATLAB-SDOT

MATLAB-SDOT is a package of **MATLAB** functions for solving the **S**emi-**D**iscrete **O**ptimal **T**ransport problem. In particular, it computes the optimal transport cost between the Lebesgue measure and a discrete measure on a 3D box with respect to the quadratic cost or the periodic quadratic cost. It implements the damped Newton method from

Jun Kitagawa, Quentin Mérigot, Boris Thibert, Convergence of a Newton algorithm for semi-discrete optimal transport. J. Eur. Math. Soc. 21 (2019), no. 9, pp. 2603–2651.

To be precise, consider the following:

- **Domain:** Let $\Omega = [0, l_1] \times [0, l_2] \times [0, l_3]$;
- **Source measure:** Let $\mu = \chi_\Omega$ be the restriction of the Lebesgue measure to Ω;
- **Target measure:** Let $\nu = \sum_{i=1}^N m_i \delta_{x_i}$ be a discrete measure, where $x_i \in \Omega$, $x_i \neq x_j$ if $i \neq j$, $m_i > 0$ and $\sum_{i=1}^N m_i = |\Omega|$, where $|\Omega|$ denotes the volume of Ω; we refer to x_i as seeds;
- **Cost:** Let $c : \Omega \times \Omega \to [0, \infty)$ be the cost function. MATLAB-SDOT includes the standard quadratic cost $c(x, y) = \|x - y\|^2$ and the periodic quadratic cost $c(x, y) = \|x - y\|_{\text{per}}^2 = \min_{k \in \mathbb{Z}^3} \|x - y - (k_1 l_1, k_2 l_2, k_3 l_3)\|^2$.

Recall that the optimal transport cost between μ and ν is defined by

$$\mathcal{T}_c(\mu, \nu) = W_2^2(\mu, \nu) = \min \left\{ \int_\Omega c(x, T(x))\, dx \ : T : \Omega \to \Omega, T\#\mu = \nu \right\}.$$

Let $w = (w_1, \dots, w_N) \in \mathbb{R}^n$ be a **weight vector.** The c**-Laguerre tessellation** of Ω generated by the seeds $x_1, \dots, x_N$ and weights $w_1, \dots, w_N$ is the partition of Ω by the sets $L_1^c, \dots, L_N^c$ defined by

$$L_i^c(w) = \{x \in \Omega : c(x, x_i) - w_i \leq c(x, x_j) - w_j \ \forall\, j \in \{1, \dots, N\}\}.$$

It is well known that the optimal transport cost is given by

$$\mathcal{T}_c(\mu, \nu) = \sum_{i=1}^N \int_{L_i^c(w_*)} c(x, x_i)\, dx = \max_{w \in \mathbb{R}^N} g(w) = g(w_*)$$

where

$$w_* = \operatorname{argmax}_{w \in \mathbb{R}^N} g(w)$$

and where $g : \mathbb{R}^N \to \mathbb{R}$ is the (concave) Kantorovich function

$$g(w) = \sum_{i=1}^N \int_{L_i^c(w)} [c(x, x_i) - w_i]\, dx + \sum_{i=1}^N m_i w_i.$$

The optimal transport map T is the piecewise-constant function given in terms of the c-Laguerre cells by $T(x) = x_i$ if $x \in L_i^c(w_*)$. In other words, every point in the cell L_i^c is transported to x_i, for each i.

The `MATLAB-SDOT` function `kantorovich` computes g and its gradient and Hessian. The function `SDOT_damped_Newton` maximises g (finds w_*) using the damped Newton method of Kitagawa, Mérigot & Thibert (2019). The function `SDOT_fminunc` maximises g using the in-built MATLAB function `fminunc` (the BFGS method), which is significantly slower than the damped Newton method.

Example 1: Quadratic cost

First we define the box $\Omega = [0,2] \times [0,3] \times [0,2.5]$:

```
l1=2; l2=3; l3=2.5; % side lengths of the box
Omega=[l1,l2,l3];
```

Next we define the target measure $\nu = \sum_{i=1}^N m_i \delta_{x_i}$. We choose the seeds x_i and the masses m_i at random:

```
N=250; % number of Dirac masses
m=rand(N,1);
m=m*l1*l2*l3/sum(m); % normalise m so that sum(m)=l1*l2*l3=volume(Omega)
X=rand(N,3)*diag(Omega); % coordinates of the seeds, randomly located in Omega
```

Here `m` is an `Nx1` array storing the masses m_i, and `X` is an `Nx3` array storing the coordinates of the seeds x_i. All the seeds must be inside the box Ω.

We specify that the cost is the standard quadratic cost c by setting

```
periodic=false;
```

Next we set the parameters of the damped Newton method:

```
w0=zeros(N,1); % initial guess for w*
percent_tol=0.1; % percentage error
```

Here `w0` is the initial guess for w_*. For the convergence of the damped Newton method w_0 must satisfy the condition $|L_i^c(w_0)| > 0$ for all i, i.e., the c-Laguerre cells $L_i^c(w_0)$ must all have non-zero volume. The damped Newton method uses this initial guess to produce an approximation w_{approx} of w_* satisfying

$$100 \times \max_i \frac{||L_i^c(w_{\text{approx}})| - m_i|}{m_i} < \texttt{percent\_tol}.$$

In other words, the c-Laguerre cells $L_i^c(w_{\text{approx}})$ have volumes m_i up to `percent_tol` percentage error.

Finally, we run the damped Newton method as follows:

```
tic
[w,exiterr,v,EXITFLAG] = SDOT_damped_Newton(w0,X,m,Omega,periodic,percent_tol);
toc
```

```
Elapsed time is 1.028231 seconds.
```

The first output argument `w` is an `N x 1` array storing the approximation w_{approx} of w_*.

The second output argument `exiterr` equals

$$100 \times \max_i \frac{||L_i^c(w_{\text{approx}})| - m_i|}{m_i}.$$

```
exiterr
```

```
exiterr = 0.0641
```

Let's check that this is less that the desired tolerance `percent_tol`:

```
disp(exiterr<percent_tol) % This should display logical 1
```

```
   1
```

The third output argument `v` is an `N x 1` array storing the actual volumes of the Laguerre cells $L_i^c(w_{\text{approx}})$:

```
i=1; % cell index
actual_volume_cell_i=v(i) % actual volume of Laguerre cell i
```

```
actual_volume_cell_i = 0.1043
```

```
target_volume_cell_i=m(i) % target volume of Laguerre cell i
```

```
target_volume_cell_i = 0.1043
```

```
percentage_error=100*(abs(actual_volume_cell_i-target_volume_cell_i))/target_volume_cell_i
```

```
percentage_error = 4.0929e-11
```

The forth output argument `EXITFLAG` can take the value `0` or `1`: `0` means that there is at least one zero-volume cell, $|L_i^c(w_{\text{approx}})| = 0$ for some i; `1` means that the algorithm was successful.

1.1 Computing the transport cost

We can compute the optimal transport cost $\mathcal{T}_c(\mu, \nu)$ using the function `mexPD` from the `MATLAB-Voro` GitHub repository:

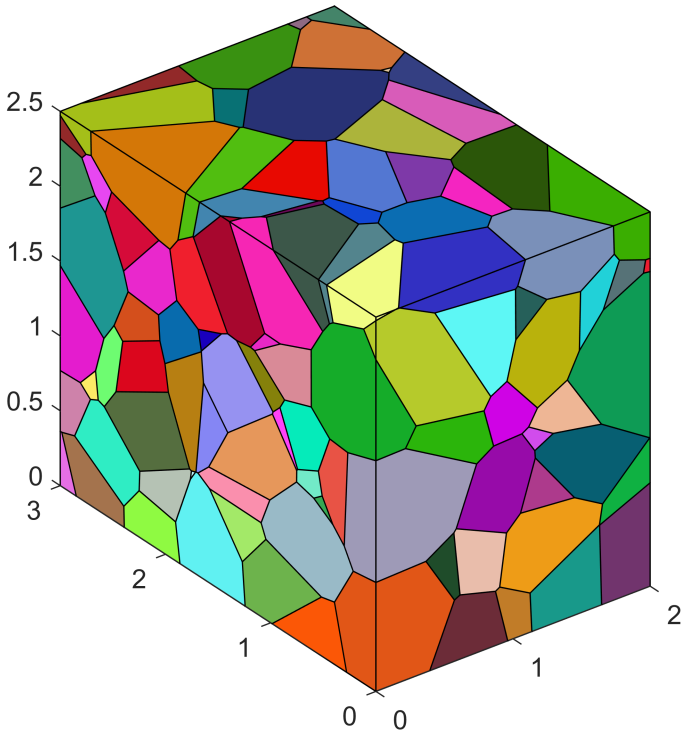
```
[~,t,~]=mexPD(Omega,X,w,periodic); % PD in mexPDall stands for 'power diagram'
transport_cost=sum(t)
```

```
transport_cost = 1.5444
```

1.2 Plotting the transport map

We can plot the Laguerre cells that represent the optimal transport map using the functions `mexPDall` and `plot_cells` from the `MATLAB-Voro` GitHub repository:

```
[~,~,~,vfn]=mexPDall(Omega,X,w,periodic);
figure
tic; plot_cells(vfn,[1:N]',rand(N,3)); toc
```




```
Elapsed time is 13.122357 seconds.
```

Warning: The plotting code is slow if the number of cells is large.

1.3 Comparison of damped Newton and BFGS

In this example we illustrate the speed of the damped Newton method compared to the BFGS method.
We start by setting up the optimal transport problem between the Lebesgue measure on the unit cube $[0,1]\times[0,1]\times[0,1]$ and the uniform discrete measure $\frac{1}{N}\sum_{i=1}^{N}\delta_{x_i}$ with $N=5000$ and randomly chosen seeds x_i:

```
Omega=[1,1,1]; % unit cube
N=5000; % number of Dirac masses
X=rand(N,3); % coordinates of the seeds
m=ones(N,1)/N; % m_i=1 for all i
periodic=false; % standard quadratic cost
w0=zeros(N,1); % initial guess for w*
percent_tol=1; % percentage error for the volumes of the cells
```

First we solve the optimal transport problem using the damped Newton method:

```
tic
[wNewton,exiterr,v,EXITFLAG] = SDOT_damped_Newton(w0,X,m,Omega,periodic,percent_tol);
toc
```

```
Elapsed time is 9.538020 seconds.
```

Next we solve the optimal transport problem using `SDOT_fminunc`, which uses the MATLAB function `fminunc` (BFGS as default):

```
tic
[wBFGS,exiterr,v] = SDOT_fminunc(w0,X,m,Omega,periodic,percent_tol);
toc
```

```
Elapsed time is 94.834118 seconds.
```

Let's check that both algorithms give the same answer (up to a constant vector):

```
norm(wNewton-wBFGS-mean(wNewton-wBFGS),Inf)
```

```
ans = 2.2925e-05
```

1.4 A large example: $N = 100,000$

Finally, we illustrate the damped Newton method on a large example. Warning: this may take ~10 minutes to run.

```
Omega=[1,1,1]; % unit cube
N=100000; % number of Dirac masses
X=rand(N,3); % coordinates of the seeds
m=ones(N,1)/N; % m_i=1 for all i
periodic=false; % standard quadratic cost
w0=zeros(N,1); % initial guess for w*
percent_tol=0.1; % percentage error for the volumes of the cells
tic
[w,exiterr,v,EXITFLAG] = SDOT_damped_Newton(w0,X,m,Omega,periodic,percent_tol);
toc
```

```
Elapsed time is 482.243093 seconds.
```

Example 2: Periodic quadratic cost

To solve the optimal transport problem with the periodic quadratic cost $c(x,y) = \|x-y\|_{\text{per}}^2$ set the `periodic` flag to `true`. For example,

```
l1=pi; % period in the x-direction
l2=3; % period in the y-direction
l3=2; % period in the z-direction
Omega=[l1,l2,l3]; % period cell
N=10000; % number of seeds
m=rand(N,1); % target volumes
m=m*l1*l2*l3/sum(m); % normalise m so that sum(m)=l1*l2*l3=volume(Omega)
X=rand(N,3)*diag(Omega); % coordinates of the seeds, randomly located in Omega
periodic=true; % periodic quadratic cost
w0=zeros(N,1); % initial guess for w*
percent_tol=0.01; % percentage error for the volumes of the cells
tic
[w,exiterr,v,EXITFLAG] = SDOT_damped_Newton(w0,X,m,Omega,periodic,percent_tol);
toc
```

```
Elapsed time is 73.139825 seconds.
```

```
[~,t,~]=mexPD(Omega,X,w,periodic);
transport_cost=sum(t) % transport cost
```

```
transport_cost = 0.1480
```