Preliminaries. Solve the following with `Matlab`, `Python`, `Mathematica` or `C/C++` (if you want to use another language, please check with me first). For the visualizations either use available functions (Matlab, Python Matplotlib or Mathematica) or write the required information to a text file which can be visualized e.g. with gnuplot. When you are done, submit the full code (and plot images) to me via email. We will then meet for a brief discussion of your work. Working in groups is encouraged.

1 Problem setup

- The graph will be encoded by a matrix $A \in \mathbb{R}^{V \times E}$ with

$$A_{x,e} = \begin{cases} 1 & \text{if } e = (x,y) \text{ for some } y \in V, \\ -1 & \text{if } e = (y,x) \text{ for some } y \in V, \\ 0 & \text{else,} \end{cases}$$

  and a vector $L \in \mathbb{R}_+^E$ which gives the length of each edge. Note that A encodes the divergence operation.

(i) For positive integers $m, n \in \mathbb{Z}_+^2$ write a function that generates A and L for a $m \times n$ Cartesian grid graph with edge length 1. The ordering of vertices and edges is up to you.

(ii) For a positive integer $m \in \mathbb{Z}_+$ write a function that generates A and L for a cyclic chain graph with edge length 1: For $i = 1, \ldots, m-1$ there is an edge from vertex i to $i+1$. Finally, there is an edge from m to 1.

2 Implementation

(i) For given A and L implement $\text{Prox}_{\tau\mathcal{C}} : \mathbb{R}^E \to \mathbb{R}^E$ and $\text{Prox}_{\sigma\theta} : \mathbb{R}^V \to \mathbb{R}^V$ where τ and σ can be given as additional parameters.

(ii) Implement iterations (3, first problem sheet) where initial iterates $(\omega^{(0)}, \phi^{(0)})$, step-sizes σ and τ and the number of total iterations ℓ are given as arguments and the final iterates $(\omega^{(\ell)}, \phi^{(\ell)})$ are returned.

(iii) Since $\omega^{(\ell)}$ will in general not satisfy $\operatorname{div} \omega^{(\ell)} = \nu - \mu$, the primal objective will strictly always be $+\infty$ during optimization (similarly, the dual score will be $-\infty$ in this problem instance). So the primal dual gap cannot be used as numerical stopping criterion.

Instead, implement a variant of the above iteration, where some $\varepsilon > 0$ is given as additional argument and the iterations run until $\| \operatorname{div} \omega^{(\ell)} - (\nu - \mu) \|_V < \varepsilon$ where $\nu - \mu$ also given as parameter and the number of required iterations is returned as additional result. (It may be prudent to implement a maximal number of allowed iterations after which the algorithm terminates with an error message.)

3 Numerical experiments

(i) For the graphs constructed in 1(i) with $m = n = 10$ set $\mu = \delta_{(1,1)}$ and $\nu = \delta_{(10,10)}$. For $(\omega^{(0)}, \phi^{(0)}) = (0^E, 0^V)$ generate a plot of $\mathcal{C}(\omega^{(\ell)})$, $\theta^*(\phi^{(\ell)})$ and $\| \operatorname{div} \omega^{(\ell)} - (\nu - \mu) \|_V$ over the number of iterations ℓ until the algorithm has approximately converged ('approximately': set the error bound $\varepsilon = 10^{-3}$). The plot of the error is probably best displayed in log scale. Plot the final iterate ϕ as a 2d function over the Cartesian grid.

(ii) Repeat the above experiment for $m = n = 5$ and $\nu = \delta_{(5,5)}$. Compare the required number of iterations until approximate convergence is achieved.

(iii) For the graph constructed in 1(ii) with $m = 10$ set $\mu = \delta_1$ and set $\nu = \delta_i$ for every $i = 1, \ldots, m$. Each time, run the algorithm until approximate convergence ($\varepsilon = 10^{-3}$). Plot the values of $\mathcal{C}(\omega)$ for the final iterate over i. Is this what you expect?