OPTIMIZATION METHODS FOR W2 GRADIENT FLOWS

ALEX ALBORS JUEZ, HAMZA GOLUBOVIC, JONATHAN FRIESEN

Department of Applied Mathematics, University of Washington, Seattle, WA
aalbors@uw.edu, ghamza@uw.edu, jfries2@uw.edu

Abstract. In this paper, we present and evaluate the different proximal splitting techniques proposed and developed in Papadakis et al's Optimal Transport with Proximal Splitting [5]. We explain the numerical scheme and more thoroughly articulate the optimization theory used in the development of multiple proximal splitting algorithms. An implementation of such algorithms is applied through a series of novel benchmarks, evaluating their performances.

1. Introduction and Overview

Optimal transport is a well-established theory concerning the transportation of probability distributions. In recent years it has found a plethora of applications in fields such as imaging, economics, geometry, and machine learning. However, the computational cost associated to much of its methods becomes prohibitive in practice. One of the most famous results concerning the numerical solution of optimal transport maps is the one due to Benamou and Brenier's seminal paper [1], which transforms the transport problem into a convex variational problem. This opened up the possibility of computing optimal transport distances through techniques of computational fluid mechanics and convex optimization, in particular the theory of proximal algorithms. In this paper we study this connection, developing the proximal mapping theory for Benamou and Brenier's objective, as well as bench-marking such algorithms through a variety of applications. Furthermore, an alternative point of view allows us to consider more general kind of objectives, leading to the computational solution to interpolated optimal transport distances between the L^2 Wasserstein and H^{-1} dual Sobolev spaces.

We will analyze the performance of two variants of the Douglas-Rachford splitting [4] and a Primal-Dual algorithm [3] for solving the discretization of the Benamou Brenier optimal transport problem. We also exemplify how all variants can be derived from the more general theory of monotone operator splitting methods [7]. We remark that most of this exposition follows the paper of Papadaki's et al [5] and the monographs of Boyd [6, 7] on operator splitting methods.

2. Theoretical Background

2.1. **Benamou Brenier formulation of OT.** In the same manner as in Papadakis et al's paper [5], we restrict our attention to smooth transport maps $T : [0,1]^d \to [0,1]^d$. In the seminal paper of Benamou-Brenier [1], it was shown that solving the L^2 optimal transport distance between two probability measures with respective densities $f^0, f^1 : \mathbb{R}^d \to \mathbb{R}_{>0}$, given by

$$\min_{T:T_{\#f^0}=f^1} \int_{[0,1]^d} |x - T(x)|^2 f^0(x) dx$$

was equivalent to the fluid mechanics nonconvex problem over densities $f(x,t) \in \mathbb{R}$ and velocity fields $v(x,t) \in \mathbb{R}^2$

$$\min_{(v,f)\in\mathcal{C}^0} \frac{1}{2} \int_{[0,1]^d} \int_0^1 f(x,t)\|v(x,t)\|^2 dt dx$$

Date: July 2, 2024.

with

$$\mathcal{C}_0 = \{\, (v, f);\ \partial_t f + \operatorname{div}_x(vf) = 0,\ v(0,\cdot) = v(1,\cdot) = 0,\ f(\cdot,0) = f^0,\ f(\cdot,1) = f^1 \,\}$$

We may now introduce the change of variables $(v, f) \mapsto (vf, f) = (m, f)$ to make the problem convex, obtaining

$$\begin{aligned} &\underset{(m,f)\,\in\,\mathcal{C}}{\text{minimize}} && \int_{[0,1]^d} \int_0^1 J(m,f)\, dt\, dx \\ &\text{subject to} && J(m,f) = \begin{cases} \frac{\|m\|^2}{2f} & \text{if } f > 0, \\ 0 & \text{if } (m,f) = (0,0)\ , \\ \infty & \text{otherwise} \end{cases} \\ & && (m,f) \in \mathcal{C}_0 \end{aligned} \tag{1}$$

Where the set $\mathcal{C}_0$ becomes

$$\mathcal{C}_0 = \{\, (m, f);\ \partial_t f + \operatorname{div}_x(m) = 0,\ m(0,\cdot) = m(1,\cdot) = 0,\ f(\cdot,0) = f^0,\ f(\cdot,1) = f^1 \,\}$$

Following [5], we discretize the problem over multidimensional grids and explore the performance of convex optimization algorithms. In particular, due to the high number of variables, the resulting discretized function is amenable to proximal splitting algorithms.

2.2. **Problem discretization.**

2.2.1. *Centered and Staggered Grids.* The integral objective 1 can be discretized with two grids over the space-time space $(x,t) \in [0,1]^d \times [0,1]$. The first one is a centered grid $\mathcal{G}_c$ which discretizes each space and time axis with $N+1$ and $P+1$ time points, respectively, for a total of $(N+1)^d \times (P+1)$ points. In the following exposition we restrict our case to $d = 1$ for simplicity. It reads

$$\mathcal{G}_c = \{(x_i = i/N,\ t_j = j/P) \in [0,1]^2\ ;\ 0 \le i \le N, 0 \le j \le P\}$$

Furthermore, by interpolating the points of the centered grid along each axis we introduce two additional staggered grids $\mathcal{G}_s^t$, $\mathcal{G}_x^t$ that allow for a more accurate computation of the divergence constraints present in the set $\mathcal{C}$. These are defined as

$$\begin{aligned} \mathcal{G}_s^x &= \{(x_i = (i+1/2)/N, t_j = j/P)\ ;\ -1 \le i \le N, 0 \le j \le P\}\,, \\ \mathcal{G}_s^t &= \{(x_i = i/N, (t_j = j+1/2)/P)\ ;\ 0 \le i \le N, -1 \le j \le P\} \end{aligned}$$

We denote $\mathcal{E}_c = (\mathbb{R}^{d+1})^{\mathcal{G}_c}, \mathcal{E}_s = \mathbb{R}^{\mathcal{G}_s^x} \times \mathbb{R}^{\mathcal{G}_s^t}$ to be the values the discretized variables (m, f) take on the centered and staggered grids, respectively. To make a distinction between them, we will explicitly denote $(\overline{m}, \overline{f}) \in \mathcal{E}_s$ to indicate they live on the staggered grid.

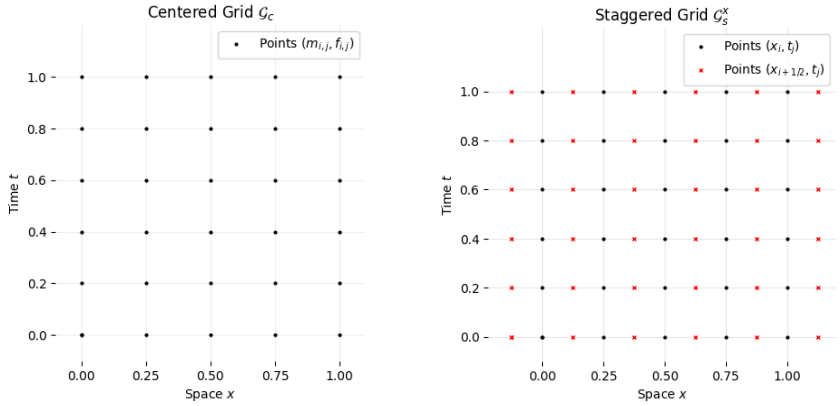


FIGURE 1. Centered and Staggered Grids

2.2.2. *Discretized operators.* We introduce an interpolation operator $\mathcal{I} : \mathcal{E}_s \to \mathcal{E}_c$ to go from the values used to compute the set constraints to the values used to evaluate the objective function.

$$\mathcal{I}(U)_{ij} = \begin{cases} m_{i,j} & = (\bar{m}_{i-1,j} + \bar{m}_{i,j})/2, \\ f_{i,j} & = (\bar{f}_{i,j-1} + \bar{f}_{i,j})/2. \end{cases} \quad \text{for} \quad 0 \le i \le N, 0 \le j \le P$$

Finally, it remains to discretize the operators used to evaluate the $\mathcal{C}_0$ set constraints, for the boundary condition we introduce $b(U) : \mathcal{E}_s \to \mathbb{R}^{2(P+1)} \times \mathbb{R}^{2(N+1)}$

$$b(U) = \left((\bar{m}_{-1,j}, \bar{m}_{N,j})_{j=0}^{P}, (\bar{f}_{i,-1}, \bar{f}_{i,P})_{i=0}^{N} \right)$$

and the space-time divergence operator

$$\operatorname{div}(U)_{i,j} = N(\bar{m}_{i,j} - \bar{m}_{i-1,j}) + P(\bar{f}_{i,j} - \bar{f}_{i,j-1}).$$

We encapsulate all $\mathcal{C}_0$ conditions by the linear map A given by

$$AU = (\operatorname{div}(U), b(U)) \quad \text{and} \quad y = (0, 0, f^0, f^1)$$

so that $AU = y$.

2.2.3. *Discretized objective function.* Having introduced all necessary notation, we may minimize over the variables in the staggered grid and evaluate the main objective by interpolating them. This amounts to the following:

$$\min_{U \in \mathcal{E}_s} \sum_{k \in \mathcal{G}_c} J(\mathcal{I}(U)_k) \;\; + \;\; \iota_{\mathcal{C}}(U)$$

where $\mathcal{C} = \{U \in \mathcal{G}_s \,;\, \operatorname{div}(U) = 0 \;\text{ and }\; b(U) = b_0\}$ represents the continuity equation and boundary conditions. The issue with this formulation comes from the need to interpolate our staggered grid variables before evaluating the objective, so we apply a common trick done when optimizing via proximal algorithms by introducing a redundant variable $V \in \mathcal{E}_c$ on the centered grid. We may now evaluate the objective with this variable and impose the condition that $\mathcal{I}(U) = V$. This amounts to the equivalent problem:

$$\min_{(U,V) \in (\mathcal{E}_s, \mathcal{E}_c)} \sum_{k \in \mathcal{G}_c} J(V) + \iota_{\mathcal{C}}(U) + \iota_{\mathcal{C}_{s,c}}((U, V))$$

with the new set constraint $\mathcal{C}_{s,c} = \{z = (U, V) \in \mathcal{E}_s \times \mathcal{E}_c : V = \mathcal{I}(U)\}$.

Having established these, we introduce an interesting extension of the objective where we consider geodesics interpolated the L^2-Wasserstein and the H^{-1} space. The generalized cost includes positive weights $(w_k)_{k \in \mathcal{G}_c} \ge 0$ and $\beta \ge 0$. It reads

$$\mathcal{J}_\beta^w(V) = \sum_{k \in \mathcal{G}_c} w_k J_\beta(m_k, f_k) \tag{2}$$

where

$$\forall\, (m, f) \in \mathbb{R}^d \times \mathbb{R}, \qquad J_\beta(m, f) = \begin{cases} \frac{\|m\|^2}{2 f^\beta} & \text{if } f > 0 \\ 0 & \text{if } (m, f) = (0, 0) \\ +\infty & \text{otherwise} \end{cases}$$

and we restrict the case $\beta \in [0, 1]$ to ensure convexity [5]. Having concluded the problem formulation, we dedicate a section to discuss the ideas surrounding the solution to optimization problems of this kind. We focus our analysis on a proximal splitting known as Douglas Rachford, which encapsulates a wide variety of well-known algorithms such as the ADMM and Primal Dual algorithms. Before introducing those, we derive the basic Douglas-Rachford algorithm and show how such a simple iterative rule can give rise to a wide variety of more complex, non-trivial optimization methods.

2.3. **Proximal splitting algorithms.** In their most general form, proximal algorithms are designed to tackle non-differentiable objectives that are compositions of simpler functions. They are often highly parallelizable and scale well with high dimensions. We will focus on the case of an objective of two functions $f + g$ that are individually tractable. We will soon see that Benamou-Brenier's formulation can be easily adapted to take such a form. Thus, let us consider the problem

$$\underset{x \in \mathcal{H}}{\text{minimize}} \quad f(x) + g(x) \tag{3}$$

where $\mathcal{H}$ is a Hilbert space. The base operation involved in such algorithms is the *proximal operator* or *proximal mapping*, which itself involves solving a smaller optimization problem.

Definition 1. *The proximal mapping of a function $f : \mathcal{H} \to \mathbb{R}$ is*

$$\mathrm{Prox}_f(x) = \arg\min_{u \in \mathcal{H}} \; f(u) + \frac{1}{2}\|x - u\|^2$$

The proximal operator of f at x seeks to minimize $f(u)$ while ensuring u is close to x. It enjoys a wide variety of properties which often make it easy to compute. For the sake of brevity we summarize them here but point to well-known references for their proofs: they may all be found in §2 of [6]

Proposition 1. *A vector $x^\star \in \mathcal{H}$ minimizer of a convex function f if and only if $0 \in \partial f(x^\star)$.*

Proposition 2. *Given a proper closed convex function f, the equality $\mathrm{Prox}_f(x) = (I + \partial f)^{-1}$ holds.*

Another identity, known as *Moreau's identity*, will be crucial in Section 3, and to state it we must introduce the following definition first.

Definition 2 (Fenchel conjugate)**.** *The Fenchel conjugate of a function f is*

$$f^*(y) = \sup_x \langle x, y \rangle - f(x)$$

We may now formulate Moreau's identity, which generalizes the orthogonal decomposition of a vector.

Proposition 3 (Moreau's identity)**.** *Given a convex function f and scalar $\lambda > 0$, the identity $x = \lambda \,\mathrm{Prox}_{f/\lambda}(x/\lambda) + \mathrm{Prox}_{\lambda f^*}(x)$ holds.*

Finally, we state two propositions which constitute the very basis of the Douglas-Rachford algorithm.

Proposition 4. *$0 \in \partial(f + g)(x) = \partial f(x) + \partial g(x)$ if and only if $(2\,\mathrm{Prox}_f - I) \circ (2\,\mathrm{Prox}_g - I)z = z$ where $x = \mathrm{Prox}_g(z)$.*

Indeed, if we can find a fixed point of the operator $C = (2\,\mathrm{Prox}_f - I) \circ (2\,\mathrm{Prox}_g - I)$, then $x = \mathrm{Prox}_g(z)$ minimizes the function $f + g$. The upshot of this result is that it allows one to work with the proximal operator of the *individual* functions instead of the entire sum, and reduces the optimization to a fixed point one. Another common result in the literature states that C is L-Lipschitz with $L = 1$ whenever f and g are convex. In the case $L < 1$, this is straightforward by applying a Banach fixed point argument: the sequence $z_{k+1} = Cz_k$ converges to a fixed point $z_k^\star$, but this need not be the case for $L = 1$. For instance, rotation maps or the operator $-I$ are both isometric but the proposed fixed point iteration doesn't converge unless $z_0 = 0$.

The key insight, first presented by Douglas and Rachford in [4], shows that if we perform the same iteration with the averaged operator $\frac{1}{2}(I + C)$, convergence is ensured whenever C has fixed points. Lastly, note this procedure is fine since the average operator has the same fixed points at C. Letting $x_{k+1} = \mathrm{Prox}_g(z_k)$, the iterative procedure $z_{k+1} = \frac{1}{2}(I + C)z_k$ reads

$$\begin{cases} x_{k+1} &= \mathrm{Prox}_g(z_k), \\ z_{k+1} &= z_k + \mathrm{Prox}_f(2x_{k+1} - z_k) - x_{k+1}. \end{cases}$$

This is known as the Douglas-Rachford algorithm. We end the section by mentioning two related algorithms, namely the Alternating Direction Method of Multipliers (ADMM) and a Primal-Dual Algorithm. We do not derive them but remark they can both be derived from a Douglas-Rachford-like iteration. A thorough dicussion of ADMM and its connection to DR can be found in section §4.4 of [6]. By adding the preconditioning term of to the objective $f+g$ and applying ADMM, we may recover the following Primal Dual algorithm. A thorough discussion can be found in §4.3 of [3]. The primal dual algorithm is applicable to objectives of the form $f+g\circ A$ where A is a linear map, and its iterations produce a sequence (w_k, x_k, y_k) of variables from an initial (x_0, y_0) according to

$$\begin{cases} y_{k+1} &= \mathrm{Prox}_{\sigma g^*}(y_k + \sigma A x_k) \\ w_{k+1} &= \mathrm{Prox}_{\tau f}(w_k - \tau A^* y_{k+1}) \\ z_{k+1} &= w_{k+1} + \theta(w_{k+1} - w_k) \end{cases}$$

where θ, τ and σ are hyperparameters. As proved in [3], if $\theta \in [0,1]$ and $\sigma\tau\|A\|^2 \leq 1$, then convergence $w_k \to w^\star$ is ensured, where $w^\star$ minimizes $f+g\circ A$.

3. Algorithm Implementation and Development

Now that we have established the theoretical foundation for proximal splitting, we aim to implement the Douglas-Rachford and Primal Dual algorithms for the Benamou-Brenier formulation. We begin by looking at the Douglas-Rachford algorithm (DR). One of the appealing properties of DR splitting is the amount of roles the functions f and g may play depending on the objective at hand. Indeed, there are many possible splitting formulations. We focus on two of them.

3.1. Assysmetric Douglas-Rachford Splitting.

Remembering the form of formulation, we can split them up as

$$\min_{(U,V)\in\mathcal{E}_\mathcal{C}\times\mathcal{E}_S} \underbrace{\mathcal{J}(V) + \iota_\mathcal{C}(U)}_{f} + \underbrace{\iota_{C_{S,C}}(U,V)}_{g}$$

Notice here that f is a separable fucntion with proximal operator $\mathrm{Prox}_{\gamma f}(U,V) = (\mathrm{Proj}_\mathcal{C}(U), \mathrm{Prox}_{\gamma\mathcal{J}}(V))$, and $\mathrm{Prox}_{\gamma g}(U,V) = \mathrm{Proj}_{\mathcal{C}_{S.C}(U,V)}$, which in calculating the projection we see,

$$(U^*, V^*) = \arg\min_{(\tilde{U},\tilde{V})} \|\tilde{U} - U\|^2 + \|\mathcal{I}\tilde{U} - V\|^2 = (I + \mathcal{I}^*\mathcal{I})^{-1}(U + \mathcal{I}^*(V))$$
$$\Rightarrow \mathrm{Prox}_{\gamma g}(U,V) = \mathrm{Proj}_{\mathcal{C}_{S,C}}(U,V) = (U^*, \mathcal{U}^*)$$

Calculating a closed form for the first two proximal operators proves to be less trivial. They will be dealt with individually.

Proposition 5. *One has*

$$\forall\, V \in \mathcal{E}_c, \quad \mathrm{Prox}_{\gamma\mathcal{J}}(V) = (\mathrm{Prox}_{\gamma J}(V_k))_{k\in\mathcal{G}_c}$$

where, for all $(\tilde{m}, \tilde{f}) \in \mathbb{R}^d \times \mathbb{R}$,

$$\mathrm{Prox}_{\gamma J}(\tilde{m}, \tilde{f}) = \begin{cases} (\mu(f^\star), f^\star) & \text{if } f^\star > 0 \\ (0,0) & \text{otherwise} \end{cases}$$

where $\forall f \geq 0$, $\mu(f) = \frac{f\tilde{m}}{f+\gamma}$ *and* $f^\star$ *is the largest real root of the third order polynomial equation in* X*:*

$$P(X) = (X - \tilde{f})(X+\gamma)^2 - \frac{\gamma}{2}\|\tilde{m}\|^2$$

The proof of this theorem is given by Theorem 4.2.1 in Papadakis et al [5]. It involves finding the global optimum of J via the subgradient condition (1) of strictly convex functions.

Proposition 6. $\mathrm{Proj}_\mathcal{C}(U) = A^*(AA^*)^{-1}y + (I - A^*(AA^*)^{-1}A)x$

Proof. We seek to project $\tilde{U} \in \mathcal{E}_S$ onto the continuity equation constraint set, encoded by $C := \{U \in \mathcal{S} : AU = y\}$. This set is an affine space which may be decomposed by $C = X + a = \{U : AU = 0\} + \{A^{-1}y\}$. Note that A^{-1} is not defined everywhere, but to avoid trivial scenarios it must be defined for y. It is easy to show that a projection onto such a set can be reduced to an operation involving the projection onto the kernel of A. Namely, $\text{Proj}_X(x) = a + \text{Proj}_X(x - a) \in C$, where we denote $u^* = \text{Proj}_X(x)$:

$$\begin{aligned}\|x - u^*\|_2^2 &= \|(x - a) - \text{Proj}_X(x - a)\|_2^2 \\ &\leq \|x - a - u\|_2^2 = \|x - v\|, \forall v \in C\end{aligned}$$

Start by noticing that $x - \text{Proj}_X(x) \perp \ker(A)$ so that $x - \text{Proj}_X(x) \in \text{Ran}(A^*)$, and so $x - \text{Proj}_X(x) = A^*z$ for some $z \in \mathcal{H}$. But then $Ax - A\,\text{Proj}_X(x) = Ax = (AA^*)z$, and so $z = (AA^*)^{-1}Ax$, and thus in total,

$$\begin{aligned}\text{Proj}_C(x) = a + \text{Proj}_X(x - a) &= A^{-1}y + (I - A^*(AA^*)^{-1}A)(x - A^{-1}y) \\ &= A^*(AA^*)^{-1}y + (I - A^*(AA^*)A)\end{aligned}$$

□

3.2. **Douglas-Rachford Updates for Benamou Brenier.** Remembering the form of the DR updates, we bring this all together to summarize our algorithm as:

$$\begin{cases} x_{k+1} &= \text{Prox}_{\iota_{C_{s,c}}(U,V)}(z_k), \\ z_{k+1} &= z_k + \alpha(\text{Prox}_{\gamma(\mathcal{J}(V)+\iota_{\mathcal{C}}(U))}(2x_{k+1} - z_k) - x_{k+1}) \end{cases}$$

with $(x^{(0)}, z^{(0)}) \in \mathcal{E}_s \times \mathcal{E}_c$ initialized arbitrarily and $\alpha \in (0, 2)$ a hyperparameter, and $\gamma > 0$. Under such conditions it is shown that $z^{(k)} \to z^*$ in [4]. Notice, however, that we can obtain a new algorithm by swapping f and g. We will call this Asymmetric-DR' (A-DR'), which defines the updates

$$\begin{cases} x_{k+1} &= \text{Prox}_{\gamma\mathcal{J}(V)+\iota_{\mathcal{C}}(U)}(z_k), \\ z_{k+1} &= z_k + \alpha(\text{Prox}_{\gamma\iota_{C_{s,c}}(U,V)}(2x_{k+1} - z_k) - x_{k+1}) \end{cases}$$

There is a bit of a tradeoff between these approaches. In the calculation of x_{k+1} in A-DR, we are at each step projecting onto the set $C_{s,c}$ on which $x_{k+1} = \mathcal{I}(z_k)$, so these two grids will be properly interpolated throughout. For A-DR' however, our x_{k+1} is calculated via a projection onto $\mathcal{C}$, so our centered variable is guaranteed to remain within the constraint set, but not necessarily properly interpolated with z_k.

3.3. **Primal Dual Updates for Benamou Brenier.** For PD, we aim to minimize functionals having the form $f + g \circ A$, for some linear operator A. In the context of Benamou Brenier, let $g = \mathcal{J}$, $A = \mathcal{I}$, and $f = \iota_{\mathcal{C}}$. Using this algorithm, we iteratively compute the sequence $(U_k, \Upsilon_k, V_k) \in \mathcal{E}_s \times \mathcal{E}_s \times \mathcal{E}_c$ via

$$\begin{cases} V_{k+1} &= \text{prox}_{\sigma\mathcal{J}^*}(V_k + \mathcal{I}(\Upsilon_k)) \\ U_{k+1} &= \text{prox}_{\tau\iota_{\mathcal{C}}^*}(V_k - \mathcal{I}^*(V_{k+1})) \\ \Upsilon_{k+1} &= U_{k+1} + \theta(U_{k+1} - U_k) \end{cases}$$

By Proposition 3, we can compute $\text{prox}_{\mathcal{J}^*}$ using $\text{prox}_{\mathcal{J}}$. Furthermore, for hyperparameters σ and θ, if $0 \leq \theta \leq 1$ and $\sigma\tau||\mathcal{I}||^2 < 1$, it can be shown that $U_k \to U^*$ [3].

3.4. **Implementation.** We implemented these algorithms with the help of the authors' MATLAB functions. The main computational cost in the proximal splitting algorithms comes from the proximal operators from Propositions 5 and 6.

Notably, the largest real root f^* in Proposition 5 is computed using the Newton–Raphson method for degree 3 polynomials. Furthermore, $(AA^*)^{-1}$ involves inverting the Laplacian, so we must solve a Poisson equation. This is computed using the Fast Fourier Transform [8] and may be done in $\mathcal{O}(N^d P \log(NP))$ operations, which constitutes the largest computational cost of all operations. Specifically, we call the multidimensional discrete cosine transform MATLAB implementation.

4. Computational Results

In this section, we will analyze the performance of the Douglas-Rachford and Primal Dual algorithms against each other on four examples. Our goal is to determine if there is a clear advantage to use one of these algorithms over the others. We note for the reader that hyperparameter tuning across the three algorithms was conducted using the Gaussian mixture model. These hyperparameters were then used across all examples presented.

Since our original objective function is $\min_{U\in\mathcal{E}_S} \mathcal{J}(\mathcal{I}(U)) + \iota_{\mathcal{C}}(U)$, we will quantify convergence in terms of $\mathcal{J}(V)$ (as $\iota_{\mathcal{C}}$ is trivial and only requires the algorithm to stay within a convex set). Therefore, we will plot the convergence with respect to the $\mathcal{J}(V)$ value at each iteration. We will also include a discussion about the convergence in violating the constraint defined by $\mathcal{C}$. Papadaki's et al [5] use this approach in quantifying convergence. However, they also consider the approach of running one of the algorithms for a very large number of iterations to obtain an optimal f^*, which they treat as the true solution, plotting $||f^* - f^{(l)}||$ at each iteration l.

4.1. **Gaussian Mixture Model.** In this example, we consider an initial density f^0 defined by a two-dimensional Gaussian distribution, and a final desnity f^1 defined by a Gaussian mixture distribution. In Figure 2, we plot the estimated densities $f_\beta(\cdot, t)$ after 2000 iterations of PD, interpolating between $f^0 = f_\beta(\cdot, 0)$ and $f^1 = f_\beta(\cdot, 1)$. In Figure 3, we evaluate the performance of the PD, A-DR, and A-DR' algorithms on the $\beta = 1$ instance (which corresponds to the W2 metric as our cost function).

As mentioned, we used the hyperparameters for PD, A-DR, and A-DR' for the $\beta = 1$ test. For PD, we used the hyperparameters in [5], given by $\sigma = 95, \tau = \frac{0.99}{\sigma||\mathcal{I}||^2}$. Optimal hyperparamters for A-DR and A-DR' were chosen using grid search. Specifically, we obtained $\gamma = 1/80$ and $\alpha = 1.75$.

The interpolation between f^0 and f^1 is plotted in Figure 2 for $\beta = 0, 1/2, 1$. Using the formulation in Section 2.2.3, note that $\beta = 1/2$ represents an interpolation between $\beta = 0$ and $\beta = 1$. The case $\beta = 0$ corresponds to using an L^2 cost function in the dynamical optimal transport formulation of estimating the path between two densities, whereas $\beta = 1$ corresponds to using the Wasserstein L^2 distance function. The plot demonstrates the power of the W2 metric, as preserving the diffusive nature when interpolating between two densities, while the regular L2 distance constructs a transport map that does not have smooth translation or dilation. It should be noted that a mix of these two metrics is sometimes desired in some applications, whereby one would use $\beta = 1/2$. Such an example is found in weather forecasting [2].

From Figure 3, we observe that the A-DR and A-DR' algorithms initially perform better, but PD eventually outperforms at roughly 600 iterations. Observe, however, that all three algorithms start with an initial spike. This is due to the implementation of the algorithms, which initialize their first iteration as a linear interpolation between densities f^0 and f^1. In other words, the first iteration is irrelevant to the proximal splitting algorithms; it is simply to jump start the process.

We also include the divergence violation plot for this example to demonstrate the nature of the three algorithms. Specifically, the value we return at each iteration of PD and A-DR' will obey the $\text{div}(U) = 0$ constraint, as discussed in Section 3.2. However, this step is reversed for A-DR, which

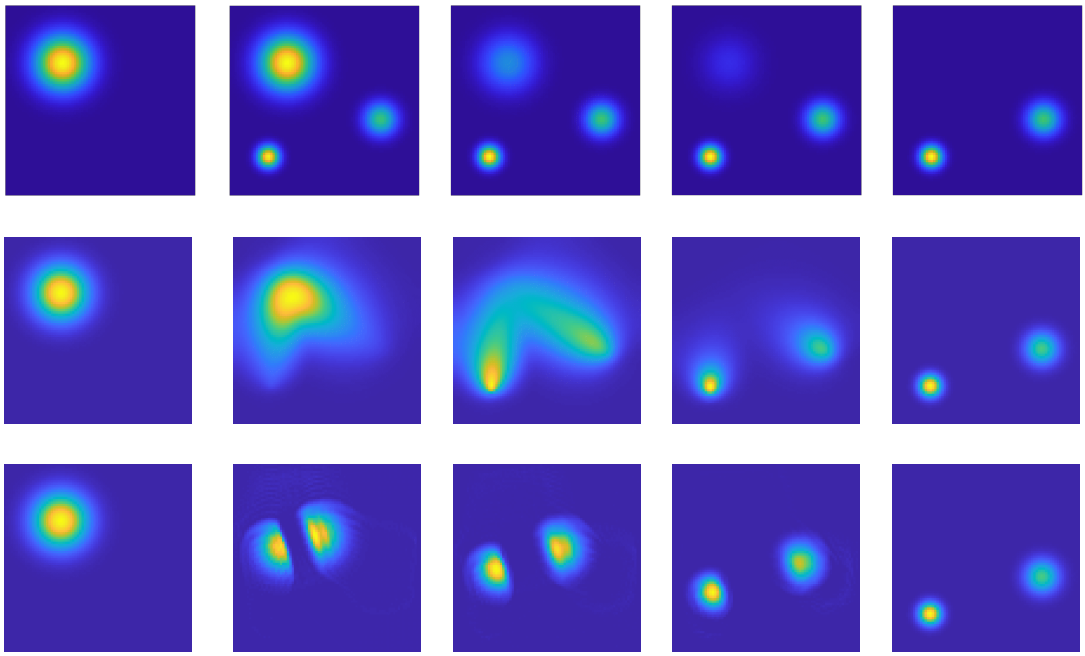


FIGURE 2. Interpolation of $f_\beta(\cdot, t)$ for the Gaussian mixture model. First row corresponds to $\beta = 0$, second row is $\beta = 1/2$, and third row is $\beta = 1$. Generated by running PD for 2000 iterations.

is why we have a non-zero violation, which eventually converges. The divergence plot is left out in the following examples, but has this same form throughout.

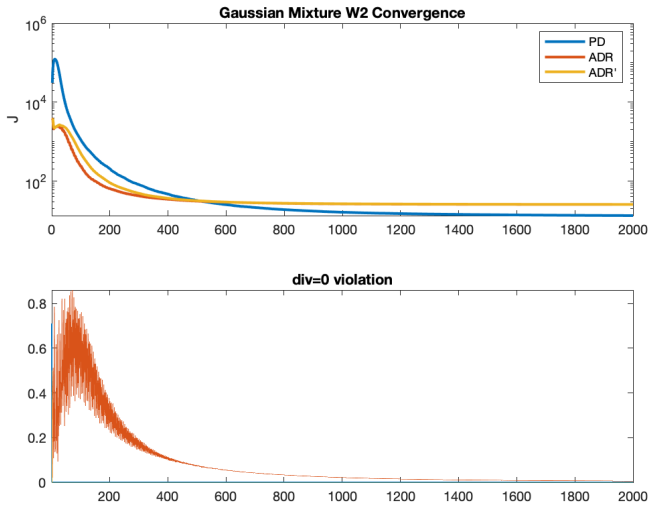
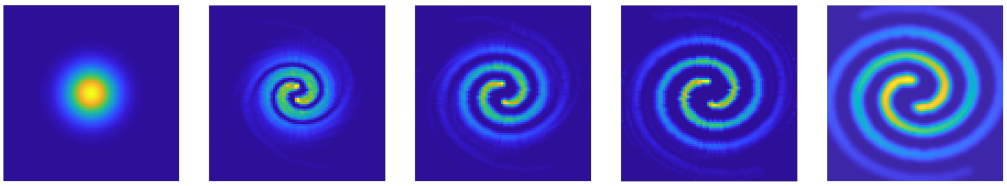


FIGURE 3. Semilog plot of the convergence for PD, A-DR, and A-DR' over 2000 iterations on the Gaussian mixture model (for $\beta = 1$). Second plot demonstrates the divergence constraint violation.

4.2. **Swirl Distribution.** In this example, we analyze the behavior of the three implemented algorithms going from a Gaussian density f^0 to a swirl distribution f^1. From Figure 5, we see that PD outperforms the DR algorithms for a majority of the iterations. However, AD-R' eventually overtakes PD in minimizing the objective function at 1800 iterations.



$t = 0$ $t = 1/4$ $t = 1/2$ $t = 3/4$ $t = 1$

FIGURE 4. Interpolation of $f_{\beta=1}(\cdot, t)$ for the swirl distribution. Generated by running PD for 2000 iterations.

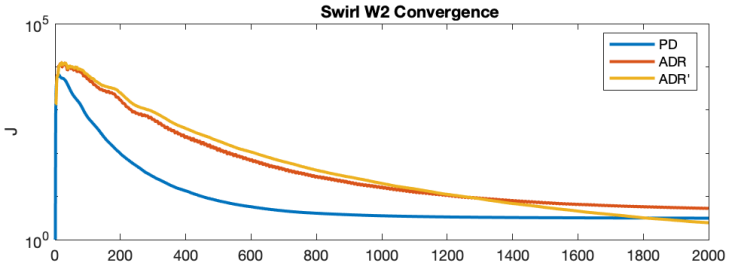
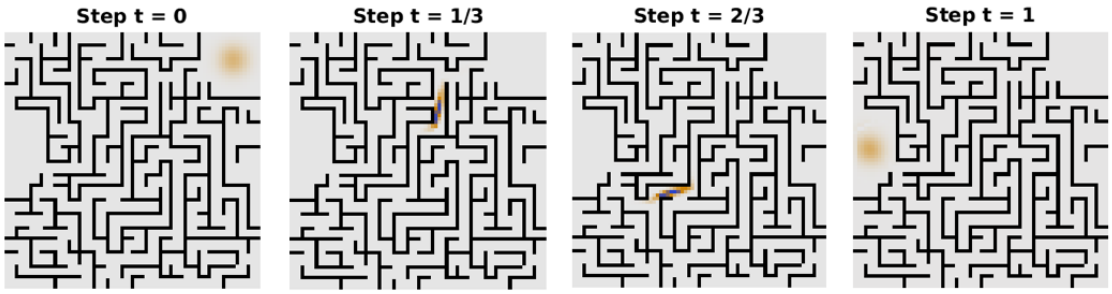


FIGURE 5. Semilog plot of the convergence for PD, A-DR, and A-DR' over 2000 iterations on the swirl distribution example (for $\beta = 1$).

4.3. **Maze.** Remembering the generalized loss function $\mathcal{J}_\beta^w$ (2) with positive weights on $\mathcal{G}_c$, if we allow specific weights to take on ∞, then we have created a boundary with which the calculated transport path will never interact with. Notice that as these weights are placed over all of $\mathcal{G}_c$, they may fluctuate with time, but we implemented a maze with fixed boundaries.

Notice from Figure 6 that this maze solver does actually find the fastest route between the start and end of the maze, never passing through boundaries, but cutting corners as efficiently as possible. Looking at our loss plot, this time our PD implementation performs better than either DR implementation for the duration of our experiment.

Furthermore, we noticed a noticeable difference in runtime. The PD algorithm took approximately half the time to run the same number of iterations as compared to the DR algorithms. We hypothesize that this comes down to the number of proximal operators computed at each iteration. DR requires calculating $\text{Prox}_J, \text{Proj}_{\mathcal{C}}$ and $\text{Proj}_{\mathcal{C}_{s,c}}$, while PD only requires calculating $\text{Prox}_{J^*}, \text{Prox}_{\iota^*}$, which can be found via Moreau's Identity.



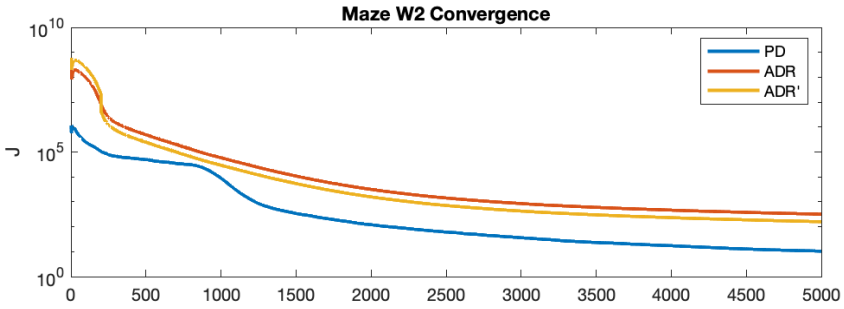


FIGURE 6. W_2 approximation ($\beta = 1$) of geodesic over the maze. Visualization given by ADR implementation. Below, we plot the semilog loss convergence for PD, A-DR, and A-DR' over 5000 iterations

5. Summary and Conclusions

From the simulations produced in Section 4.1, we see that it is difficult to summarise a clean conclusion about the performances of these algorithms. From the benchmark examples, PD outperformed the A-DR algorithms in the case of the Gaussian mixture model and maze. However, in the swirl example, we see that PD stagnates and A-DR' eventually overtakes it in finding an optimal value. It is difficult to say whether one algorithm is clearly advantageous to the others, especially in light of sensitive hyperparameter dependence and the number of iterations appropriate for each example. The complexity of our problem grows exponentially with increasing the number of grid points. For this reason, one might prefer time over accuracy.

Acknowledgements

We are thankful for Professor Jingwei's AMATH 590 lecture notes and the code packages provided by Papadakis et al in [5].

References

[1] J.-D. Benamou and Y. Brenier. A computational fluid mechanics solution to the monge-kantorovich mass transfer problem. *Numerische Mathematik*, 84:375–393, Jan 2000.

[2] J.-D. Benamou and Y. Brenier. Mixed l^2-wasserstein optimal mapping between prescribed density functions. *Journal of Optimization Theory and Applications*, 111(2):255–271, 2001.

[3] A. Chambolle and T. Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145, 2011.

[4] J. Douglas and H. H. Rachford. On the numerical solution of heat conduction problems in two and three space variables. *Transactions of the American Mathematical Society*, 82:421–439, 1956.

[5] N. Papadakis, G. Peyré, and E. Oudet. Optimal transport with proximal splitting. *SIAM Journal on Imaging Sciences*, 7(1):212–238, Jan. 2014.

[6] N. Parikh and S. Boyd. *Proximal Algorithms*. 2013. Foundations and Trends in Optimization, Vol. 1, No. 3, pp. 127-239.

[7] E. K. Ryu and S. Boyd. Primer on monotone operator methods. *Appl. Comput. Math.*, 15(1):3–43, January 2016.

[8] W. T. Vetterling, W. H. Press, S. A. Teukolsky, and B. P. Flannery. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 3rd edition, 2007.