FAST SINKHORN I: AN \(O(N)\) ALGORITHM FOR THE WASSERSTEIN-1 METRIC

QICHAEN LIAO\(^\dagger\), JING CHEN\(^\dagger\), ZIHAO WANG\(^\ddagger\), BO BAI\(^\S\), SHI JIN\(\parallel\), AND HAO WU\(^{\ast\ast}\)

Abstract. The Wasserstein metric is broadly used in optimal transport for comparing two probabilistic distributions, with successful applications in various fields such as machine learning, signal processing, seismic inversion, etc. Nevertheless, the high computational complexity is an obstacle for its practical applications. The Sinkhorn algorithm, one of the main methods in computing the Wasserstein metric, solves an entropy regularized minimizing problem, which allows arbitrary approximations to the Wasserstein metric with \(O(N^2)\) computational cost. However, higher accuracy of its numerical approximation requires more Sinkhorn iterations with repeated matrix-vector multiplications, which is still unaffordable. In this work, we propose an efficient implementation of the Sinkhorn algorithm to calculate the Wasserstein-1 metric with \(O(N)\) computational cost, which achieves the optimal theoretical complexity. By utilizing the special structure of Sinkhorn’s kernel, the repeated matrix-vector multiplications can be implemented with \(O(N)\) times multiplications and additions, using the Qin Jiushao or Horner’s method for efficient polynomial evaluation, leading to an efficient algorithm without losing accuracy. In addition, the log-domain stabilization technique, used to stabilize the iterative procedure, can also be applied in this algorithm. Our numerical experiments show that the newly developed algorithm is one to three orders of magnitude faster than the original Sinkhorn algorithm.

Keywords. Optimal Transport, Wasserstein-1 metric, Sinkhorn algorithm, FS-1 algorithm, fast algorithm

AMS subject classifications. 49M25; 49M30; 65K10

1. Introduction

The Wasserstein metric has been widely used in optimal transport for the global comparison between probabilistic distributions. It has been successfully used in various fields such as machine learning [17, 28], image processing [38], inverse problems [8, 14, 19, 33, 51], and density function theory [7, 12, 21]. Many numerical methods have been developed, including the linear programming methods [26, 36, 50], combinatorial methods [42], solving Monge-Amphère equations [4, 5, 15, 16] and proximal splitting methods [11, 34]. In recent years, several approximation techniques in optimal transport for high-dimensional distributions have also been proposed approximately [31, 32].

One of the popular numerical techniques to compute the Wasserstein metric, is the the Sinkhorn algorithm [13, 45], which minimizes the entropy regularized problem. It provides the solution roughly in \(O(N^2)\) operations with guaranteed convergence [29]. With the help of GPU acceleration, the efficiency of using the Sinkhorn algorithm to solve the optimal transport problem can be more significantly improved [25, 40, 41]. For
the Wasserstein-2 metric, the computation can be accelerated using the Gaussian convolution by approximating the geodesic distance-based kernel with the heat kernel [46]. Moreover, through statistical sampling, dimensional reduction, and other approximation methods, the complexity of the Sinkhorn algorithm can be reduced to $O(N \log N)$ [2,23] or even $O(N)$ [43]. An alternative way is to define the Wasserstein metric on the finite tree space, and the computational complexity could be $O(N \log N)$ [24,47]. Because of these progresses, the Sinkhorn algorithm has been widely used in practical problems. However, those techniques provide just approximations of the transportation cost, rather than the precise computation of the original optimization problem. Therefore, it is still of great interest to develop fast and accurate Sinkhorn-type algorithms for solving large-scale optimal transport problems.

In recent years, some fast algorithms for accurately solving the optimal transport problem have been developed recently. For example, through multi-level grids, the computation complexity can be reduced to $O(N^{1.5} \log N)$ [30]. Another well-known conclusion is that for the one-dimensional quadratic Wasserstein metric, the complexity of the sorting-based algorithm is only $O(N \log N)$ [37].

In this work, by observing the special structure of the kernel matrix in the Sinkhorn algorithm to solve the Wasserstein-1 metric, we propose a novel matrix-vector multiplication based on dynamic programming [22]. During each iteration step, it involves only a forward and backward recursive sweeping process, as in Qin Jiushao’s (or Horners’, although it appeared several centuries later) method for polynomial evaluations [20,35], which reduces the computational cost from $O(N^2)$ to $O(N)$ in each step, thus achieving the optimal theoretical complexity. In addition, the log-stabilization [9], an important technique to improve the numerical stability of the Sinkhorn algorithm, can be implemented in this strategy with the same stability property. In this paper we abbreviate this method as FS-1.

The rest of the paper is organized as follows. In Section 2, the basics of the Wasserstein-1 metric and the Sinkhorn algorithm are briefly introduced. After showing the elaborate structure of the obtained kernel matrix, we use it to develop the FS-1 algorithm in Section 3. We will also analyze the stability and integrate the log-stabilization technique into our FS-1 to improve the numerical stability. In Section 4, the FS-1 is generalized to higher dimension. We will provide numerical experiments to illustrate the huge efficiency advantage of our FS-1 algorithm in Section 5. Finally, we conclude the paper in Section 6.

2. The Wasserstein-1 metric and the Sinkhorn algorithm

Consider two probabilistic density functions $u(x)$ and $v(x)$ on a domain $\Omega \subset \mathbb{R}$, the Kantorovich’s formulation of Wasserstein-1 metric is defined as [48]:

\[
W(u,v) = \inf_{\gamma(x,y) \in \Gamma} \int_{\Omega \times \Omega} d(x,y)\gamma(x,y)dx dy,
\]

\[
\Gamma = \left\{ \gamma(x,y) \mid \int_{\Omega} \gamma(x,y)dy = u(x), \int_{\Omega} \gamma(x,y)dx = v(y) \right\}, \tag{2.1}
\]

where $d(x,y) = |x-y|$. The Sinkhorn algorithm proposed in [13,45] introduces an entropy term and solves the regularized problem:

\[
W_\varepsilon(u,v) = \inf_{\gamma(x,y) \in \Gamma} \int_{\Omega \times \Omega} |x-y|\gamma(x,y) + \varepsilon \gamma(x,y) \ln (\gamma(x,y)) dx dy. \tag{2.2}
\]

As for numerical realization, we consider two discretized probabilistic distributions

\[
u = (u_1, u_2, \cdots u_N), \quad v = (v_1, v_2, \cdots, v_N),
\]
on a uniform mesh grid with a grid spacing of $h$. Then the entropy regularized minimizing problem (2.2) can be discretized as

$$W_\epsilon(u, v) = \inf_{\gamma} \sum_{i=1}^{N} \sum_{j=1}^{N} (\gamma_{ij} |i - j|h + \epsilon \gamma_{ij} \ln(\gamma_{ij})),$$

and $\gamma_{ij}$ satisfies

$$\sum_{j=1}^{N} \gamma_{ij} = u_i, \quad \sum_{i=1}^{N} \gamma_{ij} = v_j, \quad \gamma_{ij} \geq 0.$$

The Lagrangian of the above equations writes

$$L(\gamma, \alpha, \beta) = \sum_{i=1}^{N} \sum_{j=1}^{N} (\gamma_{ij} |i - j|h + \epsilon \gamma_{ij} \ln(\gamma_{ij})) + \sum_{i=1}^{N} \alpha_i (\sum_{j=1}^{N} \gamma_{ij} - u_i) + \sum_{j=1}^{N} \beta_j (\sum_{i=1}^{N} \gamma_{ij} - v_j).$$

Taking the derivative of the Lagrangian with respect of $\gamma_{ij}$ directly leads to

$$\gamma_{ij} = e^{-\frac{3}{2} - \frac{1}{2}\alpha_i} K_{ij} e^{-\frac{1}{2} - \frac{1}{2}\beta_j},$$

where $K_{ij} = e^{-|i - j|h/\epsilon}$.

To avoid $\epsilon$ in the denominator, set $\phi_i = e^{-\frac{1}{2} - \frac{1}{2}\alpha_i}$ and $\psi_j = e^{-\frac{1}{2} - \frac{1}{2}\beta_j}$, we obtain

$$\phi_i \sum_{j=1}^{N} K_{ij} \psi_j = u_i, \quad \psi_j \sum_{i=1}^{N} K_{ij} \phi_i = v_j, \quad \forall i, j = 1, 2, \cdots, N.$$

Since the entries in $K = (K_{ij})$ are strictly positive, Sinkhorn’s iteration [45] can be applied to iteratively update vectors $\phi = (\phi_i)$ and $\psi = (\psi_j)$ by pointwise computation:

$$\psi^{(\ell+1)} = v \odot K^T \phi^{(\ell)}, \quad \phi^{(\ell+1)} = u \odot K \psi^{(\ell+1)},$$

in which the notion $\odot$ represents pointwise division and $\ell$ denotes the iterative steps.

**Remark 2.1.** In [9], a log-domain stabilization technique is proposed to reduce the numerical instability caused by the small parameter $\epsilon$. The idea is that when the infinite norms of $\phi$ or $\psi$ exceed a given threshold $\tau$, these two vectors will be normalized with the excessive part ‘absorbed’ in $\alpha$ and $\beta$:

$$\alpha \leftarrow \alpha + \ln(\phi^{(\ell)}), \quad \beta \leftarrow \beta + \ln(\psi^{(\ell)}), \quad \phi^{(\ell)} \leftarrow 1_N, \quad \psi^{(\ell)} \leftarrow 1_N.$$

Correspondingly, the matrix $K$ needs to be rescaled as $K \leftarrow \text{diag}(e^{\alpha/\epsilon}) \times K \times \text{diag}(e^{\beta/\epsilon})$.

**3. The FS-1 Algorithm** The key to the Sinkhorn algorithm is to iteratively update $\phi$ and $\psi$ through equation (2.4). By introducing the notation $\lambda = e^{-h/\epsilon}$, the matrix multiplication vector operation is written as

$$K \psi^{(\ell)} = \left( \begin{array}{c}
\psi_1^{(\ell)} + \lambda \psi_2^{(\ell)} + \lambda^2 \psi_3^{(\ell)} + \cdots + \lambda^{N-1} \psi_N^{(\ell)} \\
\lambda \psi_1^{(\ell)} + \psi_2^{(\ell)} + \lambda \psi_3^{(\ell)} + \cdots + \lambda^{N-2} \psi_N^{(\ell)} \\
\lambda^2 \psi_1^{(\ell)} + \lambda \psi_2^{(\ell)} + \psi_3^{(\ell)} + \cdots + \lambda^{N-3} \psi_N^{(\ell)} \\
\vdots & \ddots & \ddots & \vdots \\
\lambda^{N-1} \psi_1^{(\ell)} + \lambda^{N-2} \psi_2^{(\ell)} + \lambda^{N-3} \psi_3^{(\ell)} + \cdots + \psi_N^{(\ell)}
\end{array} \right).$$

(3.1)
We separate the summation of row $k$ to the lower triangular part $p_k$ and the strictly upper triangular part $q_k$. Then updating $\phi^{(\ell+1)}$ is formulated as

$$\phi_k^{(\ell+1)} = u_k/(p_k + q_k), \quad p_k = \sum_{i=1}^{k} \psi_i^{(\ell)} \lambda^{k-i}, \quad q_k = \sum_{i=k+1}^{N} \psi_i^{(\ell)} \lambda^{i-k}, \quad k = 1, 2, \ldots, N.$$  

Instead of directly calculating $p_k$ and $q_k$, we use the recursive computation given by

$$p_1 = \psi_1^{(0)}, \quad p_{k+1} = \lambda p_k + \psi_{k+1}^{(\ell)}, \quad k = 1, 2, \ldots, N-1,$$

$$q_N = 0, \quad q_k = \lambda(q_{k+1} + \psi_{k+1}^{(\ell)}), \quad k = N-1, N-2, \ldots, 1. \quad (3.2)$$

This is the Qin Jiushao or Horner method for efficient polynomial evaluation [20, 35]. Thus we develop the FS-1 algorithm with linear computational complexity, which only takes $2(N-1)$ times additions and multiplications for the matrix multiplication operation. The pseudo-code is presented in Algorithm 1.

**Algorithm 1 FS-1 Algorithm**

**Input:** $u$ and $v$ of size $(N, 1)$, $h$, $\varepsilon$, tol, itr_max

**Output:** $W_\varepsilon(u, v)$

1. $\lambda = e^{-h/\varepsilon}$; $\phi^{(0)}, \psi^{(0)} = \frac{1}{N} 1_N; p^{(0)}, r^{(0)}, q^{(0)}, s^{(0)} = 0_N; \quad \ell = 0$

2. **while** ($\ell < \text{itr max}$) and ($\sum_{i=1}^{N} |\psi_i^{(\ell)} (\sum_{j=1}^{N} \phi_j^{(\ell)} \lambda^{i-j}) - v_i | > \text{tol}$) **do**

3. $r_1^{(\ell)} = \phi_1^{(\ell)}, \quad s_N^{(\ell)} = 0$

4. **for** $i = 1 : N - 1$ **do**

5. $r_{i+1}^{(\ell)} = \lambda r_i^{(\ell)} + \phi_{i+1}^{(\ell)}$

6. $s_{N-i}^{(\ell)} = \lambda(s_{N-i+1}^{(\ell)} + \phi_{N-i+1}^{(\ell)})$

7. $\psi^{(\ell+1)} = v \odot (p^{(\ell)} + s^{(\ell)})$

8. $p_1^{(\ell)} = \psi_1^{(\ell+1)}, \quad q_N^{(\ell)} = 0$

9. **for** $i = 1 : N - 1$ **do**

10. $p_{i+1}^{(\ell)} = \lambda p_i^{(\ell)} + \psi_{i+1}^{(\ell+1)}$

11. $q_{N-i}^{(\ell)} = \lambda(q_{N-i+1}^{(\ell)} + \psi_{N-i+1}^{(\ell+1)})$

12. $\phi^{(\ell+1)} = u \odot (p^{(\ell)} + q^{(\ell)})$

13. $\ell = \ell + 1$

**return** $\sum_{i,j=1}^{N} \phi_i^{(\ell)} \psi_j^{(\ell)} \lambda^{i-j} |i-j| h$

It is well-known that the Sinkhorn algorithm has stability issues [9] due to the division and the multiplication of the small parameter $\lambda^k$. Next, we need to discuss the stability of the FS-1 algorithm to ensure that its stability is not worse than the Sinkhorn algorithm.

Consider the matrix multiplication in (3.1),

$$S^\alpha = K \psi^\alpha, \quad S^\alpha = (S_1^\alpha, S_2^\alpha, \ldots, S_N^\alpha)^T, \quad \psi^\alpha = (\psi_1^\alpha, \psi_2^\alpha, \ldots, \psi_N^\alpha)^T, \quad \alpha = 1, 2.$$  

assume that

$$|\psi_k^1 - \psi_k^2| \leq \delta, \quad k = 1, 2, \ldots, N,$$

thus


\[ |S_k^1 - S_k^2| = |\lambda^{k-1}(\psi_1^1 - \psi_1^2) + \lambda^{k-2}(\psi_2^1 - \psi_2^2) + \cdots + \lambda^{N-k}(\psi_N^1 - \psi_N^2)|, \]
\[ \leq |\lambda^{k-1} + \lambda^{k-2} + \cdots + \lambda + 1 + \lambda + \cdots + \lambda^{N-k}| \delta = \left( \frac{2 - (\lambda^k + \lambda^{N-k+1})}{1 - \lambda} - 1 \right) \delta. \]

On the other hand, consider the successive computation in (3.2), an easy induction gives
\[ |p_k^1 - p_k^2| \leq \lambda |p_{k-1}^1 - p_{k-1}^2| + |\psi_k^1 - \psi_k^2| \leq \cdots \leq (1 + \cdots + \lambda^{k-1}) \delta = \frac{1 - \lambda^k}{1 - \lambda} \delta, \]
and
\[ |q_k^1 - q_k^2| \leq \lambda |q_{k+1}^1 - q_{k+1}^2| + |\psi_{k+1}^1 - \psi_{k+1}^2| \leq \cdots \leq (\lambda + \cdots + \lambda^{N-k}) \delta \leq \left( \frac{1 - \lambda^{N-k}}{1 - \lambda} - 1 \right) \delta. \]

Thus, we have
\[ |(p_k^1 + q_k^1) - (p_k^2 + q_k^2)| \leq \left( \frac{2 - (\lambda^k + \lambda^{N-k+1})}{1 - \lambda} - 1 \right) \delta, \]
that is, the FS-1 algorithm and the Sinkhorn algorithm has the same stability.

**Remark 3.1.** Similarly, the log-domain stabilization [9] technique can also be aggregated into the FS-1 algorithm. First, we need to ‘absorb’ the excessive part of vectors \( \phi \) and \( \psi \) into \( \alpha \) and \( \beta \):
\[
\alpha \leftarrow \alpha + \ln(\phi^{(l)}), \quad \beta \leftarrow \beta + \ln(\psi^{(l)}), \quad \phi^{(l)} \leftarrow 1_N, \quad \psi^{(l)} \leftarrow 1_N.
\]
Next, we have to replace lines 5-6 and 10-11 in the Algorithm 1 as
\[
5: \quad r_{i+1}^{(l)} = \lambda e^{(b_{i+1} - b_i)/\varepsilon} r_i^{(l)} + e^{(\alpha_{i+1} + b_i)/\varepsilon} \phi_{i+1}^{(l)},
\]
\[
6: \quad s_{N-i}^{(l)} = \lambda e^{(b_{N-i} - b_{N-i+1})/\varepsilon} (s_{N-i+1}^{(l)} + e^{(\alpha_{N-i+1} + b_{N-i+1})/\varepsilon} \phi_{N-i+1}^{(l)}),
\]
\[ 10: \quad p_i^{(l)} = \lambda e^{(\alpha_{i+1} - \alpha_i)/\varepsilon} p_i^{(l)} + e^{(\alpha_{i+1} + b_{i+1})/\varepsilon} \psi_{i+1}^{(l)},
\]
\[ 11: \quad q_{N-i}^{(l)} = \lambda e^{(\alpha_{N-i} - \alpha_{N-i+1})/\varepsilon} (q_{N-i+1}^{(l)} + e^{(\alpha_{N-i+1} + b_{N-i+1})/\varepsilon} \psi_{N-i+1}^{(l)}).
\]

This leads to the stabilized FS-1 algorithm with \( O(N) \) complexity.

**Remark 3.2.** In practice, the direct calculation of \( \lambda^m = e^{-mh/\varepsilon} \) and the multiplication of \( \lambda^m \) may be troublesome since \( \lambda^m \) could be very small. However, the FS-1 algorithm gets around this problem by stepwise multiplication of \( \lambda \).

4. Extension to high dimension In this section, we illustrate how the FS-1 algorithm generalizes to higher dimensions using the two-dimensional case as an example. First, consider two discretized probabilistic distributions
\[
\mathbf{u} = (u_{11}, u_{21}, \ldots, u_{N1}, u_{12}, \ldots, u_{i1j1}, \ldots, u_{NM}),
\]
\[
\mathbf{v} = (v_{11}, v_{21}, \ldots, v_{N1}, v_{12}, \ldots, v_{i2j2}, \ldots, v_{NM}),
\]
on a uniform 2D mesh of size \( N \times M \) with a vertical spacing of \( h_1 \) and a horizontal spacing of \( h_2 \), the entropy regularized 2D Wasserstein-1 metric can be discretized as the optimal value of the following minimizing problem:
\[
W_\varepsilon(\mathbf{u}, \mathbf{v}) = \inf_{\Gamma_{i1j1,i2j2}} \sum_{i1,j1,i2,j2=1}^N \sum_{i1,j1,i2,j2=1}^M (\Gamma_{i1,j1,i2,j2}(i_1-i_2|h_1+j_1-j_2|h_2) + \varepsilon \Gamma_{i1,j1,i2,j2} \ln(\Gamma_{i1,j1,i2,j2})) \label{eq:4.1},
\]

where $\Gamma_{i_1j_1i_2j_2}$ satisfies
\[
\sum_{i_2=1}^{N} \sum_{j_2=1}^{M} \Gamma_{i_1j_1i_2j_2} = u_{i_1j_1}, \quad \sum_{i_1=1}^{N} \sum_{j_1=1}^{M} \Gamma_{i_1j_1i_2j_2} = v_{i_2j_2}, \quad \Gamma_{i_1j_1i_2j_2} \geq 0.
\]

Same as in the 1D case, the problem (4.1) can be solved by the Sinkhorn iteration. If $\mathbf{u}$ and $\mathbf{v}$ are flattened into 1D vectors in column-major order, the corresponding kernel matrix is written as
\[
K = \left( \begin{array}{cccc}
K_0 & \lambda_2 K_0 & \lambda_2^2 K_0 & \cdots & \lambda_2^{M-1} K_0 \\
\lambda_2 K_0 & K_0 & \lambda_2 K_0 & \cdots & \lambda_2^{M-2} K_0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\lambda_2^{M-1} K_0 & \lambda_2^{M-2} K_0 & \lambda_2^{M-3} K_0 & \cdots & K_0
\end{array} \right),
\]
where the sub-matrix
\[
K_0 = \left( \begin{array}{cccc}
1 & \lambda_1 & \cdots & \lambda_1^{N-1} \\
\lambda_1 & 1 & \cdots & \lambda_1^{N-2} \\
\vdots & \vdots & \ddots & \vdots \\
\lambda_1^{N-1} & \lambda_1^{N-2} & \cdots & 1
\end{array} \right),
\]
and
\[
\lambda_1 = e^{-h_1/\varepsilon}, \quad \lambda_2 = e^{-h_2/\varepsilon}.
\]

Obviously, the cost of direct matrix-vector multiplication in Sinkhorn is $O(N^2M^2)$. By using the FS-1’s trick twice, we expect the computational cost can be significantly reduced, the key idea is as follows. Let
\[
\psi_i^{(\ell)} = \left( \psi_{1i}^{(\ell)}, \psi_{2i}^{(\ell)}, \cdots, \psi_{Ni}^{(\ell)} \right), \quad i = 1, 2, \ldots, M,
\]
the matrix-vector multiplication $K\psi^{(\ell)}$ is written as
\[
K\psi^{(\ell)} = \left( \begin{array}{c}
K_0\psi_1^{(\ell)} + \lambda_2 K_0\psi_2^{(\ell)} + \lambda_2^2 K_0\psi_3^{(\ell)} + \cdots + \lambda_2^{M-1} K_0\psi_M^{(\ell)} \\
\lambda_2 K_0\psi_1^{(\ell)} + K_0\psi_2^{(\ell)} + \lambda_2 K_0\psi_3^{(\ell)} + \cdots + \lambda_2^{M-2} K_0\psi_M^{(\ell)} \\
\lambda_2^2 K_0\psi_1^{(\ell)} + \lambda_2 K_0\psi_2^{(\ell)} + K_0\psi_3^{(\ell)} + \cdots + \lambda_2^{M-3} K_0\psi_M^{(\ell)} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\lambda_2^{M-1} K_0\psi_1^{(\ell)} + \lambda_2^{M-2} K_0\psi_2^{(\ell)} + \lambda_2^{M-3} K_0\psi_3^{(\ell)} + \cdots + K_0\psi_M^{(\ell)}
\end{array} \right), \quad (4.2)
\]
We separate the summation of row $k$ to the lower triangular part $\mathbf{p}_k$ and the strictly upper triangular part $\mathbf{q}_k$. Then updating $\phi^{(\ell+1)}$ is formulated as
\[
\phi_{k}^{(\ell+1)} = \mathbf{u}_k \odot (\mathbf{p}_k + \mathbf{q}_k), \quad \mathbf{p}_k = \sum_{i=1}^{k} K_0 \psi_i^{(\ell)} \lambda_2^{k-i}, \quad \mathbf{q}_k = \sum_{i=k+1}^{M} K_0 \psi_i^{(\ell)} \lambda_2^{i-k}, \quad k = 1, \ldots, M.
\]
Instead of directly calculating \( p_k \) and \( q_k \), a successive computation is used

\[
p_1 = K_0 \psi_1^{(\ell)}, \quad p_{k+1} = \lambda_2 p_k + K_0 \psi_k^{(\ell)}, \quad k = 1, 2, \ldots, M - 1, \\
q_M = 0_N, \quad q_k = \lambda_2 (q_{k+1} + K_0 \psi_{k+1}^{(\ell)}), \quad k = M - 1, M - 2, \ldots, 1.
\] (4.3)

Thus, the total cost of matrix-vector multiplication of our FS-1 algorithm for 2D Wasserstein-1 metric is reduced to \( O(NM) \). The pseudo-code is presented in Algorithm 2.

This idea can be easily extended to high-dimensional cases, and we will not repeat it here. An argument similar to the one used in Section 3 shows that the FS-1 algorithm and the Sinkhorn algorithm in the 2D case still has the same stability. Thus, we shall omit the discussions.

**Algorithm 2 2D FS-1 Algorithm**

**Input:** \( u \) and \( v \) of size \((N \times M, 1)\), \( h_1, h_2, \varepsilon, \text{tol}, \text{itr}\_max\)  
**Output:** \( W_\varepsilon(u, v) \)

```plaintext
1: \( \lambda_1 = e^{-h_1/\varepsilon}; \lambda_2 = e^{-h_2/\varepsilon}; \phi^{(0)}, \psi^{(0)} = \frac{1}{NM} 1_{NM}; p^{(0)}, r^{(0)}, q^{(0)}, s^{(0)} = 0_N; \ell = 0 \)
2: \textbf{while} \((\ell < \text{itr}\_max)\) and \((\sum_{i=1}^{M} |\psi_i^{(\ell)}| \circ (\sum_{j=1}^{M} \lambda_2 |i-j| K_0 \phi_j^{(\ell)}) - v_1| > \text{tol})\) \textbf{do}
3: \( r_1^{(\ell)} = K_0 \psi_1^{(\ell)} \) (using Algorithm 1 with \( h = h_1, \varepsilon = \varepsilon \)), \( s^{(\ell)} = 0_N \)
4: \textbf{for} \( i = 1 : M - 1 \) \textbf{do}
5: \( r_{i+1}^{(\ell)} = \lambda_2 r_i^{(\ell)} + K_0 \psi_i^{(\ell+1)} \)
6: \( s_{M-i}^{(\ell)} = \lambda_2 (s_{M-i+1}^{(\ell)} + K_0 \phi_{i+1}^{(\ell)}) \)
7: \( \psi_i^{(\ell+1)} = v \circ (r_i^{(\ell)} + s_i^{(\ell)}) \)
8: \( p_i^{(\ell)} = K_0 \psi_i^{(\ell+1)} \) (using Algorithm 1 with \( h = h_1, \varepsilon = \varepsilon \)), \( q_M^{(\ell)} = 0_N \)
9: \textbf{for} \( i = 1 : M - 1 \) \textbf{do}
10: \( p_{i+1}^{(\ell)} = \lambda_2 p_i^{(\ell)} + K_0 \psi_{i+1}^{(\ell+1)} \)
11: \( q_{M-i}^{(\ell)} = \lambda_2 (q_{M-i+1}^{(\ell)} + K_0 \psi_{M-i+1}^{(\ell+1)}) \)
12: \( \phi^{(\ell+1)} = u \circ (p^{(\ell)} + q^{(\ell)}) \)
13: \( \ell = \ell + 1 \)

\textbf{return} \( \sum_{i_1, j_1=1}^{N} \sum_{i_2, j_2=1}^{M} \phi_{i_1, j_1}^{(\ell)} \psi_{i_2, j_2}^{(\ell)} \lambda_1 |i_1-i_2| \lambda_2 |j_1-j_2| (|i_1-i_2| h_1 + |j_1-j_2| h_2) \)
```

5. Numerical Experiments  In this section, we carry out four numerical experiments to evaluate the FS-1 algorithm. The first two examples show the performance of the FS-1 algorithm in 1D cases. Specifically, we consider the comparison of two 1D random distributions and the comparison of two Ricker wavelets arising from seismology [8]. The last two examples show the performance of the FS-1 algorithm in 2D cases. Specifically, we consider the comparison of two 2D random distributions and the comparison of two images arising from the image matching problem [39]. The marginal error \( \| \text{diag}(\psi) \times K^T \times \phi - v \| \) is chosen as the termination condition [3, 44]. All the experiments are conducted on a platform with 128G RAM, and one Intel(R) Xeon(R) Gold 5117 CPU @2.00GHz with 14 cores.

5.1. 1D random distributions  We consider the Wasserstein-1 metric between two 1D random distributions on the interval \([-3, 3]\). There are uniform grid points

\[
x_i = (i-1) \Delta x - 3, \quad \Delta x = \frac{6}{N-1}, \quad i = 1, 2, \ldots, N.
\]
Correspondingly, we consider the two random vectors on the grid points
\[ u = (u_1, u_2, \ldots, u_N), \quad v = (v_1, v_2, \ldots, v_N), \]
where \( u_i \) and \( v_j \) are both uniformly distributed on \([0, 1]\). We would like to compare the performance and computational cost on computing the Wasserstein-1 metric \( W_\epsilon \left( \frac{u}{\|u\|}, \frac{v}{\|v\|} \right) \) using the Sinkhorn algorithm and the FS-1 algorithm. We tested 100 random experiments, and each experiment was performed for 1000 iterations.

In Table 5.1, we output the averaged computational time of two different algorithms. We can see that the FS-1 algorithm has an overwhelming advantage in computational speed. Moreover, the transport plans obtained by the two algorithms are almost identical. To further study the efficiency advantage of our FS-1 algorithm, we present the computational time of the two algorithms in different cases. By data fitting, we can see the empirical complexity of the FS-1 algorithm is \( O(N^{1.02}) \), while that of the Sinkhorn algorithm is \( O(N^{2.22}) \), see Figure 5.1 (Left) for illustration. In Figure 5.1 (Right), we discuss the computational time required to reach the corresponding marginal error under different regularization parameters \( \epsilon \) for the random distribution with dimension \( N = 10000 \). Obviously, the FS-1 algorithm is about two orders of magnitude faster than the Sinkhorn algorithm.

| N     | Computational time (s)             | Speed-up ratio | \( \|P_{NB} - P\|_F \) |
|-------|-------------------------------------|----------------|--------------------------|
| 500   | 8.70 \times 10^{-3}                | 7.68 \times 10^{-2} | 8.83 \times 10^0 | 6.54 \times 10^{-15} |
| 2000  | 4.25 \times 10^{-2}                | 2.81 \times 10^0  | 6.61 \times 10^1 | 4.98 \times 10^{-18} |
| 8000  | 1.53 \times 10^{-1}                | 4.80 \times 10^1  | 3.14 \times 10^2 | 3.92 \times 10^{-18} |

Table 5.1. The 1D random distribution problem. The comparison between the Sinkhorn algorithm and the FS-1 algorithm with the different number of grid points \( N \). The regularization parameter \( \epsilon = 0.001 \). Columns 2-4 are the averaged computational time of the two algorithms and the speed-up ratio of the FS-1 algorithm. Column 5 is the Frobenius norm of the difference between the transport plan computed by the two algorithms.

5.2. Ricker wavelet

Next, we consider the computation of the Wasserstein-1 metric between the Ricker wavelet
\[ R(t) = A(1 - 2\pi^2 f_0^2 t^2) e^{-\pi^2 f_0^2 t^2}, \]
and its translation \( R(t - s) \). The Ricker wavelet is commonly used to model source time function in seismology [8]. Here \( f_0 \) is the dominant frequency, and \( A \) denotes the wave amplitude. For simplicity, we set
\[ f_0 = 1, \quad A = 1. \]

Since the Ricker wavelet is not always positive over the entire time duration, we will square and normalize it for the comparisons of the Wasserstein-1. In [27], a new normalization method with better convexity is given as follows:
\[ D(f, g) = W_{1,\epsilon} \left( \frac{f^2}{\|f\|^2} + \delta \frac{g^2}{\|g\|^2} + \delta \right), \quad (5.1) \]
where $\delta$ is a sufficiently small parameter to improve numerical stability, and $L$ is a given parameter to guarantee that the two functions being compared are normalized.

Below, we randomly select the translation parameter $s = -1.2032$. This parameter can ensure that the two Ricker wavelets $R(t)$ and $R(t - s)$ are sufficiently far apart. And the small parameter $\delta = 10^{-3}$. We repeated the experiment 100 times, and each experiment was performed for 500 iterations. In Table 5.2, we output the averaged computational time of two different algorithms. We also present the computational time required to reach the corresponding marginal error under different regularization parameters $\varepsilon$ for the distribution with dimension $N = 10000$, see Figure 5.2 (Left) for illustration, from which, we can draw the same conclusions as those in Subsection 5.1.

In Figure 5.2 (Right), we also discuss the impact of the log-domain stabilization technique for $\varepsilon = 0.001$. Without the technique, the Sinkhorn algorithm terminates abnormally at the 97th iteration and the FS-1 algorithm terminates abnormally at the 104th iteration. By introducing the log-domain stabilization technique, neither algorithm is terminated abnormally. Moreover, the stabilized FS-1 algorithm still maintains a significant efficiency advantage over the stabilized Sinkhorn algorithm.

| $N$   | Computational time (s) | Speed-up ratio | $||P_{NB} - P||_F$ |
|-------|------------------------|----------------|-------------------|
| 500   | $4.90 \times 10^{-3}$  | $2.30 \times 10^{-2}$ | $4.69 \times 10^0$ | $5.67 \times 10^{-16}$ |
| 2000  | $1.52 \times 10^{-2}$  | $1.36 \times 10^0$   | $8.93 \times 10^1$ | $1.81 \times 10^{-17}$ |
| 8000  | $5.96 \times 10^{-2}$  | $2.81 \times 10^1$   | $4.72 \times 10^2$ | $1.22 \times 10^{-16}$ |

Table 5.2. The Ricker wavelet problem. The comparison between the Sinkhorn algorithm and the FS-1 algorithm with the different number of grid points $N$. The regularization parameter $\varepsilon = 0.01$. Columns 2-4 are the averaged computational time of the two algorithms and the speed-up ratio of the FS-1 algorithm. Column 5 is the Frobenius norm of the difference between the transport plan computed by the two algorithms.

5.3. 2D Random distributions Next, we discuss the performance of the FS-1 algorithm in two dimension. This Subsection generalizes the 1D random distributions in
Subsection 5.1 to 2D random distributions. The basic settings are almost the same as in Subsection 5.1. And we need to compute the Wasserstein-1 metric between two $N \times N$ dimensional random vectors, where $N$ is the number of grid points at each dimension. Without loss of generality, we set $h_1 = h_2 = 1$. We also tested 100 random experiments, and each experiment was performed for 1000 iterations. The averaged computational time for different numbers of nodes $N \times N$ of the two algorithms are output in Table 5.3 and Figure 5.3 (Left). By data fitting, we can see the empirical complexity of the FS-1 algorithm is $O(N^{1.66})$, while that of the Sinkhorn algorithm is $O(N^{4.44})$. These results are even better than the expected $O(N^2)$ complexity for the FS-1 algorithm. This again shows the big efficiency advantage of the FS-1 algorithm compared to the Sinkhorn algorithm. In Figure 5.3 (Right), we also present the computational time required to reach the corresponding marginal error under different regularization parameters $\varepsilon$ for the random distribution with dimension 100 × 100. Obviously, the FS-1 algorithm still maintains the efficiency advantage for more than two orders of magnitude.

| $N \times N$ | Computational time (s) | Speed-up ratio | $||P_{NB} - P||_F$ |
|--------------|------------------------|----------------|-------------------|
| $10 \times 10$ | $1.27 \times 10^{-2}$  | $8.02 \times 10^{-2}$ | $6.34 \times 10^0$ | $1.20 \times 10^{-17}$ |
| $20 \times 20$ | $2.01 \times 10^{-2}$  | $6.73 \times 10^{-1}$ | $3.34 \times 10^1$ | $5.96 \times 10^{-18}$ |
| $40 \times 40$ | $7.38 \times 10^{-2}$  | $1.38 \times 10^1$ | $1.87 \times 10^2$ | $3.00 \times 10^{-18}$ |
| $80 \times 80$ | $2.64 \times 10^{-1}$  | $4.78 \times 10^2$ | $1.81 \times 10^3$ | $1.55 \times 10^{-18}$ |
| $160 \times 160$ | $1.08 \times 10^0$     | $1.38 \times 10^4$ | $1.28 \times 10^4$ | $7.68 \times 10^{-19}$ |

Table 5.3. The 2D random distribution problem. The comparison between the Sinkhorn algorithm and the FS-1 algorithm with the different number of grid points $N \times N$. The regularization parameter $\varepsilon = 0.01$. Columns 2-4 are the averaged computational time of the two algorithms and the speed-up ratio of the FS-1 algorithm. Column 5 is the Frobenius norm of the difference between the transport plan computed by the two algorithms.

5.4. Image matching problem
An important application of the optimal transport in 2D is to match images. OT
Fig. 5.3. The 2D random distribution problem. Left: The comparison of computational time between the FS-1 algorithm and the Sinkhorn algorithm with different numbers of grid nodes \( N \). Right: The computational time required to reach the corresponding marginal error under different regularization parameter \( \varepsilon = 0.1 \) (Green), \( \varepsilon = 0.05 \) (Purple), and \( \varepsilon = 0.01 \) (Red).

plays a fundamental role of related tasks including density regularization [6], image registration [18], and optical flow [10]. The complexity advantage of the FS-1 algorithm compared to the Sinkhorn algorithm can further enable practical applications of optimal transport in high-resolution images.

Here we consider the image matching experiment. We randomly select two images, see Figure 5.4 for illustration, from the DIV2K dataset [1], where the images have 2K pixels for at least one of the axes (vertical or horizontal). Considering that the resolutions of the images are different, we first sample them to the same scale \( N \times N \). Without loss of generality, we set \( h_1 = h_2 = 1 \). In addition, to facilitate the optimal transport comparison, we convert them to grayscale images and normalize them using formula (5.1) with \( \delta = 10^{-7} \). Again, we repeated the experiment 100 times and each experiment was performed for 1000 iterations.

Fig. 5.4. The image matching problem. Illustration of images.

In Table 5.4, we output the averaged computational time of two different algorithms. It should be emphasized that the computational time of the Sinkhorn algorithm is too long for large-scale images, so there is no result. However, the FS-1 algorithm can still handle this simply. We also present the computational time required to reach the corresponding marginal error under different regularization parameters \( \varepsilon \). Here, the image size is 100\( \times \)100, see Figure 5.5 (Left) for illustration, from which we can draw the same conclusions as those in Subsection 5.3.

In Figure 5.5 (Right), we also discuss the impact of the log-domain stabilization
**Fig. 5.5.** The image matching problem. Left: The computational time required to reach the corresponding marginal error under different regularization parameter $\varepsilon = 1$ (Green), $\varepsilon = 0.5$ (Purple), and $\varepsilon = 0.1$ (Red). Right: The comparison between the Sinkhorn-type algorithms with (Red) and without (Blue) the log-domain stabilization for $\varepsilon = 0.01$.

technique for $\varepsilon = 0.01$. Without the technique, the Sinkhorn algorithm and the FS-1 algorithm both terminate abnormally at the 138th iteration. By introducing the log-domain stabilization technique, neither algorithm is terminated abnormally. Moreover, the stabilized FS-1 algorithm still maintains a significant efficiency advantage over the stabilized Sinkhorn algorithm.

| $N \times N$ | Computational time (s) | Speed-up ratio | $\| P_{NB} - P \|_F$ |
|-------------|------------------------|----------------|-----------------|
| 100×100     | 4.10×10^{-1} 1.32×10^3 | 3.23×10^{3} 2.28×10^{-17} |
| 200×200     | 1.72×10^0 3.08×10^4 | 1.79×10^{5} 9.52×10^{-18} |
| 400×400     | 7.23×10^0 | – | – |
| 800×800     | 3.20×10^1 | – | – |

**Table 5.4.** The image matching problem. The comparison between the Sinkhorn algorithm and the FS-1 algorithm with the different total number of grid nodes $N \times N$. The regularization parameter $\varepsilon = 1$. Columns 2-4 are the averaged computational time of the two algorithms and the speed-up ratio of the FS-1 algorithm. Column 5 is the Frobenius norm of the difference between the transport plan computed by the two algorithms.

## 6. Conclusion

In this paper, we propose an efficient (abbreviated as FS-1) algorithm to compute the Wasserstein-1 metric with linear computational cost per iteration. This method is developed by discovering the natural structure of Sinkhorn’s kernel, which allows a matrix-vector multiplication to be carried out exactly with $O(n)$ cost for each iteration by using Qin Jiushao’s or Horner’s method for efficient polynomial evaluation. Moreover, the FS-1 algorithm can also be adapted to the widely used log-domain stabilization technique. As shown by numerous experiments, the FS-1 algorithm achieves a huge speed advantage without losing accuracy.

Finally, this paper mainly considers the acceleration of the Sinkhorn algorithm in the matrix-vector multiplication. It is well known that the number of iterations of the Sinkhorn algorithm will significantly increase with the increase of numerical accuracy, which leads to slow convergence. In [49], the Inexact Proximal point method for the Optimal Transport problem (IPOT) was proposed for this problem. We believe that
FS-1 and IPOT can be effectively combined to develop a new algorithm for solving the Optimal Transport problem with fast convergence and low complexity. We are currently investigating this important extension and hope to report the progress in a future paper.

Acknowledgements. This work was supported by National Natural Science Foundation of China Grant Nos.11871297 and 12031013, Tsinghua University Initiative Scientific Research Program, and Shanghai Municipal Science and Technology Major Project 2021SHZDZX0102.

REFERENCES

[1] E. Agustsson and R. Timofte, Ntire 2017 challenge on single image super-resolution: Dataset and study, in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, July 2017. 5.4
[2] J. Altschuler, F. Bach, A. Rudi, and J. Weed, Massively scalable Sinkhorn distances via the Nyström method, Advances in Neural Information Processing Systems, 1050 (2018), p. 12. 1
[3] J. Altschuler, J. Weed, and P. Rigollet, Near-linear time approximation algorithms for optimal transport via Sinkhorn iteration, arXiv preprint arXiv:1705.09634, (2017). 5
[4] J.-D. Benamou and Y. Brenier, A computational fluid mechanics solution to the Monge-Kantorovich mass transfer problem, Numerische Mathematik, 84 (2000), pp. 375–393. 1
[5] J.-D. Benamou, B. D. Froese, and A. M. Oberman, Numerical solution of the optimal transportation problem using the Monge–Ampère equation, Journal of Computational Physics, 260 (2014), pp. 107–126. 1
[6] M. Burger, M. Franek, and C.-B. Schönlieb, Regularized regression and density estimation based on optimal transport, Applied Mathematics Research eXpress, 2012 (2012), pp. 209–253. 5.4
[7] G. Buttazzo, L. De Pascale, and P. Gori-Giorgi, Optimal-transport formulation of electronic density-functional theory, Physical Review A, 85 (2012), p. 062502. 1
[8] J. Chen, Y. Chen, H. Wu, and D. Yang, The quadratic Wasserstein metric for earthquake location, Journal of Computational Physics, 373 (2018), pp. 188–209. 1, 5, 5.2
[9] L. Chizat, G. Peyré, B. Schmitzer, and F.-X. Vialard, Scaling algorithms for unbalanced optimal transport problems, Mathematics of Computation, 87 (2018), pp. 2563–2609. 1, 2.1, 3, 3.1
[10] P. Clarysse, B. Delhay, M. Picq, and J. Pousin, Optimal extended optical flow subject to a statistical constraint, Journal of Computational and Applied Mathematics, 234 (2010), pp. 1291–1302. 5.4
[11] P. L. Combettes and J.-C. Pesquet, Proximal splitting methods in signal processing, in Fixed-Point Algorithms for Inverse Problems in Science and Engineering, Springer, 2011, pp. 185–212. 1
[12] C. Cotar, G. Friesecke, and C. Klüppelberg, Density functional theory and optimal transportation with Coulomb cost, Communications on Pure and Applied Mathematics, 66 (2013), pp. 548–599. 1
[13] M. Cuturi, Sinkhorn distances: Lightspeed computation of optimal transport, Advances in Neural Information Processing Systems, 26 (2013), pp. 2292–2300. 1, 2
[14] B. Engquist, K. Ren, and Y. Yang, The quadratic Wasserstein metric for inverse data matching, Inverse Problems, 36 (2020), p. 055001. 1
[15] B. D. Froese, Numerical methods for the elliptic Monge-Ampère equation and optimal transport, PhD thesis, Science: Department of Mathematics, 2012. 1
[16] B. D. Froese and A. M. Oberman, Convergent finite difference solvers for viscosity solutions of the elliptic Monge–Ampère equation in dimensions two and higher, SIAM Journal on Numerical Analysis, 49 (2011), pp. 1692–1714. 1
[17] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, Generative adversarial nets, Advances in Neural Information Processing Systems, 27 (2014). 1
[18] S. Haker, L. Zhu, A. Tannenbaum, and S. Angenent, Optimal mass transport for registration and warping, International Journal of Computer Vision, 60 (2004), pp. 225–240. 5.4
[19] H. Heaton, S. W. Fung, A. T. Lin, S. Osher, and W. Yin, Wasserstein-based projections with applications to inverse problems, arXiv preprint arXiv:2008.02200, (2020). 1
[45] R. Sinkhorn, Diagonal equivalence to matrices with prescribed row and column sums, The American Mathematical Monthly, 74 (1967), pp. 402–405. 1, 2, 2

[46] J. Solomon, F. De Goes, G. Peyré, M. Cuturi, A. Butscher, A. Nguyen, T. Du, and L. Guibas, Convolutional wasserstein distances: Efficient optimal transportation on geometric domains, ACM Transactions on Graphics (ToG), 34 (2015), pp. 1–11. 1

[47] Y. Takezawa, R. Sato, and M. Yamada, Supervised tree-Wasserstein distance, in Proceedings of the 38th International Conference on Machine Learning, M. Meila and T. Zhang, eds., vol. 139 of Proceedings of Machine Learning Research, PMLR, 18–24 Jul 2021, pp. 10086–10095. 1

[48] C. Villani, Optimal transport: old and new, vol. 338, Springer, 2009. 2

[49] Y. Xie, X. Wang, R. Wang, and H. Zha, A fast proximal point method for computing exact wasserstein distance, in Uncertainty in artificial intelligence, PMLR, 2020, pp. 433–453. 6

[50] L. Yang, J. Li, D. Sun, and K.-C. Toh, A fast globally linearly convergent algorithm for the computation of Wasserstein barycenters., Journal of Machine Learning Research, 22 (2021), pp. 1–37. 1

[51] Y. Yang, B. Engquist, J. Sun, and B. F. Hamfeldt, Application of optimal transport and the quadratic Wasserstein metric to full-waveform inversion, Geophysics, 83 (2018), pp. R43–R62. 1