Introduction théorique et numérique au transport optimal

Zakaria Otmane

26 mai 2025 - 30 septembre 2025

Département des Mathématiques Appliquées, INSA Rennes

Grupo de Fisica Matematica, IST Lisbonne

Projet de recherche : Analyse théorique et numérique du transport optimal - Stage de 4A

Maître de stage : Professeur Léonard Monsaingeon

Tuteur INSA : Professeur Mohamed Camar Eddine

Table des matières

Introduction

Dans le cadre de ma quatrième année à l'INSA de Rennes en Mathématiques Appliquées, j'ai réalisé un stage de recherche à l'Instituto Superior Técnico de Lisbonne, au sein du *Grupo de Física Matemática*, encadré par Léonard Monsaingeon, expert en équations aux dérivées partielles elliptiques et paraboliques non linéaires. Ses travaux portent notamment sur le transport optimal généralisé et ses applications aux EDP, les flots de gradient en espaces métriques, les équations dégénérées, la modélisation mathématique pour la diffusion en physique et biologie, les problèmes à frontière libre et les équations de réaction-diffusion. Le laboratoire, financé par la Fondation pour la Science et la Technologie du Portugal, se situe à l'interface entre mathématiques pures et physique théorique, avec des axes principaux en analyse stochastique, systèmes intégrables, physique quantique et transport optimal. J'y ai travaillé en binôme avec un étudiant de l'ENSTA Paris sur la théorie et la résolution numérique des problèmes de transport optimal.
Après une étude approfondie de la monographie *Computational Optimal Transport* de Peyré et Cuturi, j'ai implémenté et analysé l'algorithme de Sinkhorn pour le transport optimal entropiquement régularisé, en étudiant sa convergence et ses applications notamment en imagerie. Ce travail m'a permis d'acquérir des compétences solides en programmation scientifique et en modélisation mathématique.

Ce rapport est structuré de la manière suivante :

- **Chapitre 1 :** Présentation des bases mathématiques du transport optimal : rappel historique du problème de Monge, formulation moderne en espaces métriques, version relaxée de Kantorovich et principaux résultats théoriques, dont le théorème de Brenier, en vue de l'étude numérique.
- **Chapitre 2 :** Aspects numériques du transport optimal : discrétisation du problème de Kantorovich, régularisation entropique pour améliorer la stabilité, description détaillée de l'algorithme de Sinkhorn avec exemples et analyse d'erreur, puis extension aux barycentres et version stable en domaine logarithmique.
- **Chapitre 3 :** Application au traitement d'images : modélisation des images comme distributions de masse discrètes et utilisation de l'algorithme de Sinkhorn pour calculer des barycentres générant des images intermédiaires entre une image source et une image cible.

Les codes informatiques correspondant aux algorithmes évoqués dans ce rapport, ainsi que certaines démonstrations mathématiques jugées pertinentes, sont regroupés en annexes pour consultation. De plus, un lien vers un notebook détaillant davantage les programmes informatiques est également fourni dans ces annexes.

1 La théorie du transport optimal

Nous entamons cette partie par une présentation des motivations historiques ayant conduit à l'émergence de la problématique du transport optimal. Nous allons ensuite voir une formulation moderne et générale du problème, suivi de l'étude du célèbre problème de transport de Monge-Kantorovich avant de finir ensuite par quelques aspects théoriques liés aux problèmes.

1.1 Le problème historique de Monge

Le problème du transport optimal est formulé pour la première fois en 1781 par le mathématicien français Gaspard Monge, dans un *Mémoire sur la théorie des déblais et des remblais* présenté à l'Académie des Sciences. Il cherche alors à répondre à la question suivante :

Quelle est la manière la plus économique de remplir un trou avec un tas de sable ?

Dans la modélisation de Monge, le coût associé au transport d'une unité de masse depuis un point x vers un point y est donné par la distance euclidienne entre ces deux points.

Modélisons ce problème de la manière suivante : soient $A, B \subset \mathbb{R}^2$ deux sous-ensembles boréliens [1] tels que

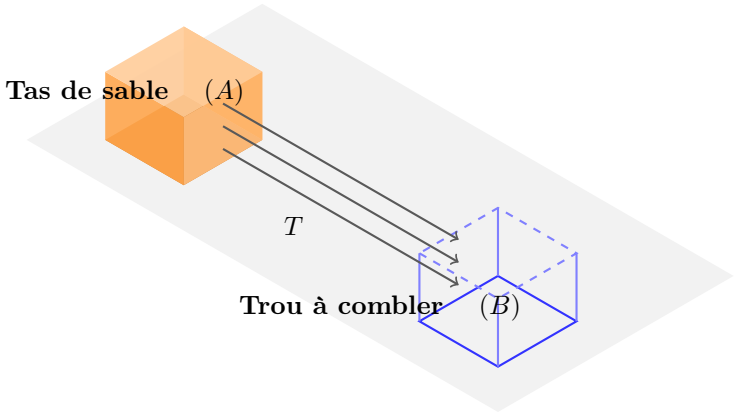
$$0 < \lambda_2(A), \lambda_2(B) < +\infty,$$

où λ_2 désigne la mesure de Lebesgue sur $\mathbb{R}^2$. On associe à ces ensembles deux mesures de probabilité α et β définies par

$$d\alpha(x) = \frac{\mathbf{1}_A(x)}{\lambda_2(A)}\,dx, \quad d\beta(x) = \frac{\mathbf{1}_B(x)}{\lambda_2(B)}\,dx.$$

Les ensembles A et B désignent respectivement la position initiale du tas de sable et celle du trou à combler. La mesure source α décrit la distribution initiale de masse (supposée uniforme) et la mesure cible β la répartition finale (également uniforme). Le problème de Monge consiste à trouver une application mesurable $T : \mathbb{R}^2 \to \mathbb{R}^2$ transportant α sur β en minimisant un coût donné.

1. On appelle tribu borélienne de $\mathbb{R}^d$, notée $\mathcal{B}(\mathbb{R}^d)$, la plus petite tribu contenant tous les ouverts de $\mathbb{R}^d$.
Un sous-ensemble borélien de $\mathbb{R}^d$ est un élément de $\mathcal{B}(\mathbb{R}^d)$.



Expliquons ce que signifie transporter α sur β. On demande à l'application T de satisfaire, pour tout ensemble borélien $C \subset \mathbb{R}^2$,

$$\alpha\left(\{x \in \mathbb{R}^2 : T(x) \in C\}\right) = \beta(C),$$

c'est-à-dire que la *mesure image* de α par T, notée $T_{\#}\alpha$, doit être égale à β. On utilise classiquement la notation $T_{\#}\alpha$ pour désigner cette mesure image, définie par

$$T_{\#}\alpha(C) := \alpha\left(T^{-1}(C)\right) = \alpha\left(\{x \in \mathbb{R}^2 : T(x) \in C\}\right).$$

T doit donc vérifier la contrainte $T_{\#}\alpha = \beta$.

Comme par la formulation de Monge, le coût associé au transport d'une unité de masse d'un point x vers son image $T(x)$ est donné par $\|x - T(x)\|$, le coût total associé à une application de transport T s'écrit alors :

$$C(T) := \int_{\mathbb{R}^2} \|x - T(x)\| \, d\alpha(x).$$

Le problème de Monge revient donc à déterminer la quantité :

$$C(\alpha, \beta) := \inf_{\substack{T : A \to B \\ T_{\#}\alpha = \beta}} C(T).$$

Toute application T réalisant cet infimum sera notée T^*.

1.2 Formulation moderne et générale du problème de Monge

Donner une formulation générale du problème de Monge permet d'élargir le cadre à des espaces plus abstraits. On considère donc deux espaces métriques polonais [2] $(\mathcal{X}, d)$ et $(\mathcal{Y}, d')$, ainsi qu'une fonction de coût

$$c : \mathcal{X} \times \mathcal{Y} \to [0, +\infty]$$

2. Il s'agit d'un espace métrique complet et séparable.

qui représente le coût de transport d'un point x vers un point y. On note $\mathcal{P}(\mathcal{X})$ et $\mathcal{P}(\mathcal{Y})$ les ensembles des mesures de probabilité boréliennes sur $\mathcal{X}$ et $\mathcal{Y}$. Une application mesurable $T : \mathcal{X} \to \mathcal{Y}$ est dite *application de transport* de la probabilité $\alpha \in \mathcal{P}(\mathcal{X})$ vers une probabilité $\beta \in \mathcal{P}(\mathcal{Y})$ si, comme dit dans la section 1.1, la mesure image de α par T est égale à β, c'est-à-dire :

$$T_{\#}\alpha = \beta.$$

Définition 1 : *Pour toute telle application T, on définit le coût total du transport par :*

$$C(T) := \int_{\mathcal{X}} c(x, T(x))\, d\alpha(x).$$

Le problème de Monge consiste alors à déterminer une application optimale minimisant ce coût parmi toutes les applications de transport admissibles, autrement dit :

$$C(\alpha, \beta) := \inf_{\substack{T:\mathcal{X}\to\mathcal{Y} \\ T_{\#}\alpha=\beta}} C(T). \tag{1}$$

Lorsque cet infimum est atteint, l'application correspondante T^* est appelée *plan de transport optimal* ou *application de Monge optimale*.

1.3 Formulation relaxée de Kantorovich

La formulation (1) peut s'avérer mal posée : en effet, il n'existe pas toujours d'application mesurable T telle que $T_{\#}\alpha = \beta$.

Pour pallier cette difficulté, une reformulation plus souple a été proposée au XX$^{\text{e}}$ siècle par Leonid Kantorovich. Elle repose sur l'idée de considérer non plus une application, mais une mesure sur le produit $\mathcal{X} \times \mathcal{Y}$, appelée *couplage*, représentant une stratégie probabiliste de transport.

Définition 2 : *Soient $(\alpha, \beta) \in \mathcal{P}(\mathcal{X}) \times \mathcal{P}(\mathcal{Y})$. On note $\mathcal{U}(\alpha, \beta)$ l'ensemble des mesures de probabilité sur $\mathcal{X} \times \mathcal{Y}$ dont les marges sont respectivement α et β* [3]*. Pour tout $\pi \in \mathcal{U}(\alpha, \beta)$, le coût total de transport associé est défini par :*

$$C(\pi) := \int_{\mathcal{X}\times\mathcal{Y}} c(x, y)\, d\pi(x, y).$$

Le problème de Kantorovich consiste alors à minimiser ce coût parmi tous les couplages admissibles :

$$\mathcal{T}_c(\alpha, \beta) := \inf_{\pi \in \mathcal{U}(\alpha,\beta)} C(\pi). \tag{2}$$

Lorsqu'un couplage $\pi^* \in \mathcal{U}(\alpha, \beta)$ réalise ce minimum, on dit que π^* est un *couplage optimal* ou un *plan de transport optimal* entre α et β.

3. C'est-à-dire que pour tout $\pi \in \mathcal{U}(\alpha, \beta)$ on a $d\alpha(x) = \int_{\mathcal{Y}} d\pi(x, y)$ et $d\beta(y) = \int_{\mathcal{X}} d\pi(x, y)$.

L'un des avantages majeurs de la formulation de Kantorovich est que, sous des hypothèses très générales sur la fonction de coût c (notamment sa continuité et sa positivité), il existe toujours au moins un couplage optimal, ce qui garantit l'existence d'une solution au problème de transport.

1.4 Quelques résultats théoriques

L'objectif de cette section est de poser des résultats théoriques de référence avant de traiter le problème d'un point de vue numérique. En particulier le **théorème de Brenier** jouera un rôle fondamental dans cette section. Le théorème s'énonce de la manière suivante.

Théorème 3 : *Soient α et β deux mesures sur $\mathbb{R}^d$, absolument continues par rapport à la mesure de Lebesgue, avec des moments d'ordre deux finis. Alors il existe une fonction convexe $\phi : \mathbb{R}^d \to \mathbb{R}$ telle que le transport optimal de α vers β, pour le coût quadratique $c(x, y) = \|x - y\|^2$, est donné par*

$$T^*(x) = \nabla\phi(x).$$

De plus, cette application est unique α-presque partout.

Sous ces conditions, le plan optimal π^* est également unique et il est concentré sur le graphe de T^*. Les deux formalismes coïncident à travers la relation :

$$\pi^*(x, y) = \alpha(x)\, \delta_{T^*(x)}(y), \tag{3}$$

où $\delta_{T^*(x)}$ désigne la mesure de Dirac en $T(x)$. Ainsi, π^* est supportée sur le graphe $(x, T^*(x))$. Aussi en adoptant une vision strictement probabiliste, alors à partir de (3), si on considère le couple aléatoire $(X, Y) \sim \pi^*$, on a :

$$\mathbb{E}[Y \mid X = x] = \int_{\mathcal{Y}} y\, \frac{d\pi^*(x, y)}{d\alpha(x)} = \int_{\mathcal{Y}} y\, d\delta_{T^*(x)}(y) = T^*(x). \tag{4}$$

Ceci nous donne une autre relation entre π^* et T^*. Cette dernière nous sera particulièrement utile pour approcher T^* avec les méthodes numériques de la section 2.

Aussi, si on applique ce théorème en **dimension 1**, toute fonction $T : \mathbb{R} \to \mathbb{R}$ qui pousse α sur β et qui est **croissante** est **optimale**. Il n'y a donc pas de croisement lorsque les matières sont transportées. En particulier, dans ce cas-ci, le théorème 3 permet de trouver une **formule théorique** permettant de déterminer T^*. En effet, pour une mesure θ sur $\mathbb{R}$ donnée et strictement positive, on note F_θ sa fonction de répartition, c'est-à-dire :

$$F_\theta(x) = \int_{-\infty}^{x} \mathrm{d}\theta(u), \quad \text{pour tout } x \in \mathbb{R}.$$

De ce fait, F_θ est inversible et on rappelle que le quantile est donné par F_θ^{-1}. Nous pouvons alors énoncer le théorème suivant :

Théorème 4 : *Le transport optimal T^* entre deux mesures réelles α et β est donné par :*

$$T^* = F_\beta^{-1} \circ F_\alpha.$$

Il sera alors possible dans certains cas de figure de déterminer explicitement T^*. Dans toute la suite nous nous placerons principalement dans le contexte du théorème 3 pour illustrer les résultats des méthodes numériques et les confronter à des résultats théoriques.

2 Résolution numérique

Les bases théoriques étant désormais bien établies, nous nous intéressons à présent aux schémas numériques permettant de résoudre de manière approximative les problèmes de transport optimal. L'objectif sera notamment de concevoir des méthodes efficaces minimisant les erreurs de calcul, tout en prenant en considération les limitations inhérentes à l'implémentation informatique sous Python.

2.1 Formulation numérique discrétisée

Nous commençons par discrétiser le problème. Nous avons vu dans la section 1.3 que la formulation (2) est celle qui est la mieux posée. C'est donc sur cette dernière que nous allons nous concentrer pour la suite. On pose : $\Sigma_n := \{\, p \in \mathbb{R}_+^n \mid \sum_{i=1}^n p_i = 1 \,\}$, l'ensemble des vecteurs de probabilités de taille n.

Nous discrétisons l'ensemble $\mathcal{X} \times \mathcal{Y}$ en un pavé de points (x_i, y_j) avec $(i,j) \in \{1,\dots,n\} \times \{1,\dots,m\}$, où $(n,m) \in (\mathbb{N}^*)^2$ sont choisis suffisamment grands. Dans ce contexte, les couplages π sur $\mathcal{X} \times \mathcal{Y}$ sont représentés par des matrices P de taille $n \times m$, telles que $P_{i,j}$ représente la masse transportée du point x_i vers le point y_j. On a donc :

$$\pi = \sum_{i=1}^{n} \sum_{j=1}^{m} P_{i,j}\, \delta_{i,j}.$$

De même, la fonction coût $c : \mathcal{X} \times \mathcal{Y} \to [0, +\infty[$ est discrétisée sous la forme d'une matrice C de taille $n \times m$, où chaque entrée $C_{i,j}$ représente le coût de transport associé au déplacement d'une unité de masse du point x_i vers le point y_j. Nos mesures discrètes s'écrivent alors de la manière suivante :

$$\alpha = \sum_{i=1}^{n} a_i \delta_i \quad \text{avec } a_i \geq 0 \ \forall i \in \{1,\dots,n\}, \quad \sum_{i=1}^{n} a_i = 1,$$

$$\beta = \sum_{j=1}^{m} b_j \delta_j \quad \text{avec } b_j \geq 0 \ \forall j \in \{1,\dots,m\}, \quad \sum_{j=1}^{m} b_j = 1.$$

Les vecteurs $a = (a_1, \ldots, a_n)^T \in \Sigma_n$ et $b = (b_1, \ldots, b_m)^T \in \Sigma_m$ représentent alors les distributions de probabilité discrètes en dimension finie. δ_i et $\delta_{i,j}$ représentent les impulsions de Dirac aux points x_i et (x_i, y_j).

Dans ce cadre, les contraintes marginales s'écrivent matriciellement :

$$P\mathbf{1}_m = a \quad \text{et} \quad P^T\mathbf{1}_n = b,$$

On cherche alors à minimiser le coût total du transport, donné par :

$$C(P) := \sum_{i=1}^{n}\sum_{j=1}^{m} C_{i,j}P_{i,j} = \langle C, P\rangle,$$

où $\langle C, P\rangle$ désigne le produit scalaire de Frobenius entre les matrices C et P.
La version discrète du problème de Kantorovich consiste donc à résoudre le problème d'optimisation suivant :

$$L_C(a,b) := \min_{P\in U(a,b)} C(P), \tag{5}$$

où l'ensemble admissible est défini par :

$$U(a,b) := \left\{P \in (\mathbb{R}_+)^{n\times m} \;\middle|\; P\mathbf{1}_m = a,\ P^T\mathbf{1}_n = b\right\}.$$

C'est un problème d'optimisation linéaire, avec une **fonction objectif linéaire sur un polytope**[4] défini par des contraintes d'égalité. L'absence de stricte convexité rend la mise au point de méthodes numériques beaucoup plus délicate vis-à-vis de la stabilité. Pour y remédier, on ajoute un terme de **régularisation** basé sur l'**entropie**, rendant la fonction strictement convexe. Cette approche, à la base du *transport optimal entropiquement régularisé*, permet des algorithmes plus stables et efficaces, comme **l'algorithme de Sinkhorn**.

2.2 Régularisation entropique et algorithme de Sinkhorn

Nous commençons par introduire la notion d'entropie dans le cadre matriciel.

Définition 5 : *Soit $P \in (\mathbb{R}_+)^{n\times m}$, on appelle entropie de la matrice P la quantité définie par :*

$$H(P) := \sum_{i=1}^{n}\sum_{j=1}^{m} P_{i,j}\left(\log(P_{i,j}) - 1\right).$$

Si l'une des composantes de P est nulle, alors on pose $H(P) := -\infty$.

4. Un polytope est un ensemble convexe formé par l'enveloppe convexe d'un nombre fini de points dans un espace euclidien $\mathbb{R}^d$. Autrement dit, c'est le plus petit ensemble convexe contenant ces points.

La fonction H ainsi définie est **strictement convexe** sur l'ensemble des matrices à coefficients strictement positifs.

L'idée est à présent de résoudre une version régularisée de (5), en introduisant un terme entropique afin de rendre la fonction objectif strictement convexe. Pour un paramètre $\varepsilon > 0$, on considère le problème suivant :

$$L_C^\varepsilon(a,b) := \min_{P \in U(a,b)} \left(C(P) + \varepsilon H(P) \right), \tag{6}$$

Le problème ainsi formulé est **strictement convexe**. Nous avons en particulier la propriété suivante.

Proposition 6 : *En notant P^ε une solution de* (6) *on a le résultat suivant,*

$$\lim_{\varepsilon \to 0} P^\varepsilon = P^*,$$

où P^* est une solution optimale de (5)*. En particulier,*

$$\lim_{\varepsilon \to 0} L_C^\varepsilon(a,b) = L_C(a,b).$$

Ainsi, en prenant ε suffisamment petit, nous sommes en mesure de résoudre de manière approchée (5). Pour un tel ε, la résolution numérique peut être effectuée à l'aide de **l'algorithme de Sinkhorn**, qui repose sur la proposition suivante.

Proposition 7 : *P^ε est unique et s'écrit sous la forme*

$$P^\varepsilon_{i,j} = u_i K_{i,j} v_j,$$

où $K_{i,j} := e^{-C_{i,j}/\varepsilon}$ et $(u,v) \in \mathbb{R}_+^n \times \mathbb{R}_+^m$. La matrice K est appelée, noyau de Gibbs.

Matriciellement, on peut réécrire P^ε sous la forme :

$$P^\varepsilon = \mathrm{diag}(u)\, K\, \mathrm{diag}(v).$$

Les contraintes marginales imposent alors :

$$\mathrm{diag}(u)\, K\, \mathrm{diag}(v)\, \mathbf{1}_m = a, \quad \text{et} \quad \mathrm{diag}(v)\, K^\top\, \mathrm{diag}(u)\, \mathbf{1}_n = b.$$

On observe aisément que ces équations s'écrivent aussi :

$$u \odot (Kv) = a, \quad \text{et} \quad v \odot (K^\top u) = b,$$

où le symbole $\odot$ désigne la multiplication terme à terme entre vecteurs.

Une manière intuitive de résoudre ces équations consiste à les satisfaire de manière itérative, en mettant à jour successivement u pour qu'elle vérifie la première équation, puis v pour qu'elle vérifie la seconde. Il s'agit du principe des

projections de Bregman. Ces deux mises à jour définissent, à la $(l+1)$-ième itération, le coeur de l'algorithme de Sinkhorn :

$$u^{(l+1)} := \frac{a}{Kv^{(l)}}, \quad v^{(l+1)} := \frac{b}{K^\top u^{(l+1)}} \tag{7}$$

où l'opérateur de division entre vecteurs est également à comprendre au sens terme à terme. Au vu des contraintes marginales, nous choisissons le critère d'arrêt suivant :

$$\texttt{err} := \|P_l^\varepsilon \mathbf{1}_m - a\|_1 < \texttt{tol}$$

Il est inutile de se concentrer sur la deuxième marginale. En effet, le principe de projection assure que :

$$\left(P_l^\varepsilon\right)^T \mathbf{1}_n = v^{(l)} \odot (K^T u^{(l)}) = \frac{b}{K^T u^{(l)}} \odot (K^T u^{(l)}) = b.$$

Nous allons à présent appliquer cette méthode à un exemple donné.

2.2.1 Un exemple illustratif simple entre deux distributions

Nous appliquons cet algorithme à un problème de transport optimal dans lequel les distributions de masse sont des fonctions uniformes de largeur 0.3, la seconde étant obtenue par translation de la première d'une valeur $\tau = 0.5$.

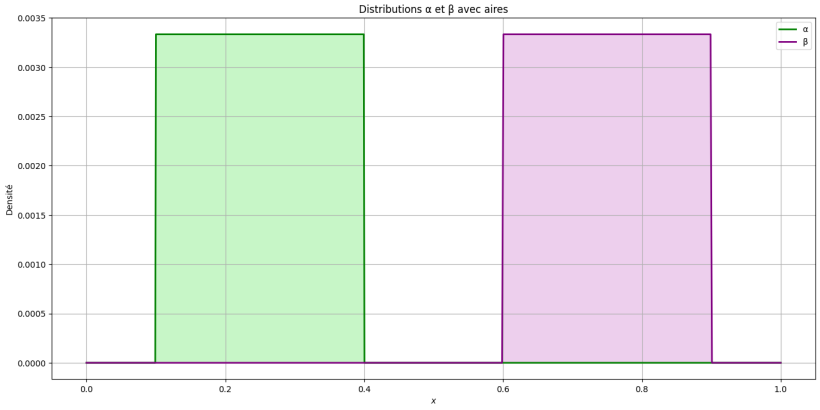


FIGURE 1 – Distributions uniformes de α et β

Théoriquement, on peut vérifier à l'aide du théorème 4 que l'application optimale est donnée par :

$$T^*(x) = x + \tau.$$

Nous appliquons l'algorithme de Sinkhorn à ce problème pour $\varepsilon \in \{1, 0.1, 0.01, 0.001\}$, afin d'observer l'effet de la régularisation entropique sur le plan de transport. Nous fixons le nombre maximal d'itérations à $\texttt{max\_iter} = 1000$ et une tolérance

`tol` $= 10^{-9}$. Le plan optimal régularisé obtenu est représenté à travers le graphe suivant.

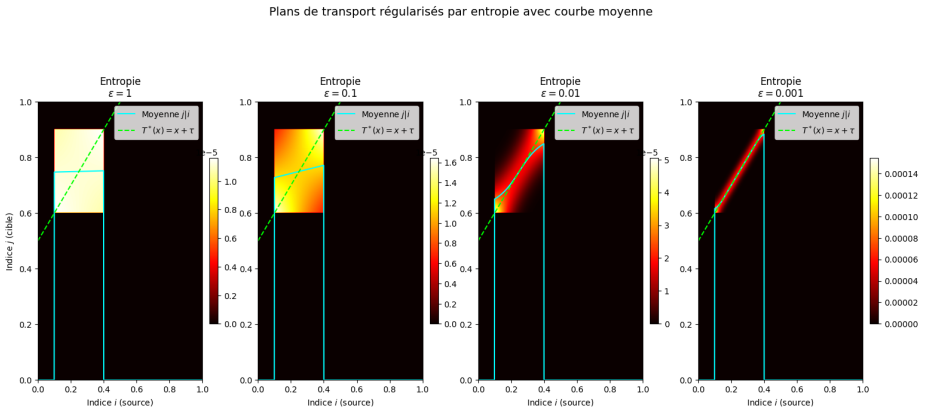


FIGURE 2 – Plans de transport entropiques

Pour des valeurs de ε de plus en plus faibles, le plan de transport optimal s'affine et la moyenne des distributions de masse obtenues coïncide de plus en plus avec la courbe théorique. Celle-ci est calculée en associant à chaque point source x_i la moyenne pondérée des points cibles y_j selon les coefficients du plan de transport régularisé P^ε. Il s'agit en fait d'une version empirique de (4) :

$$\bar{y}_i := \frac{\sum_{j=1}^{n} P^\varepsilon_{i,j} y_j}{\sum_{j=1}^{n} P^\varepsilon_{i,j}} = \frac{(P^\varepsilon y)_i}{(P^\varepsilon \mathbf{1_m})_i}$$

où y désigne le vecteur des positions cibles. Cet exemple illustre la précision de la méthode numérique basée sur l'algorithme de Sinkhorn.

Cependant, lorsque $\varepsilon \sim 10^{-4}$, le programme rencontre des problèmes numériques liés à la présence de zéros machine. En effet les mises à jour itératives (7) explosent et s'effondrent car $K = e^{-C/\varepsilon}$ devient presque nulle pour de tels ε. Ceci limite la stabilité et la fiabilité des calculs dans ce régime de faible régularisation.

Il est donc judicieux d'étudier le comportement des erreurs liées à cet algorithme.

2.2.2 Une étude sur les erreurs numériques

La théorie établit que les erreurs sur les contraintes marginales de notre algorithme, après un nombre d'itérations l suffisamment grand, sont de la forme :

$$\texttt{err} \approx \frac{1}{l^{A(\varepsilon)}},$$

à condition que ε soit suffisamment petit. Par conséquent, dans ce contexte, on a :

$$\log_{10}(\texttt{err}) \approx -A(\varepsilon)\log_{10}(l)$$

Ainsi, $A(\varepsilon)$ joue le rôle de coefficient directeur dans la relation linéaire entre $\log_{10}(\text{err})$ et $\log_{10}(l)$. Nous avons le graphe suivant :

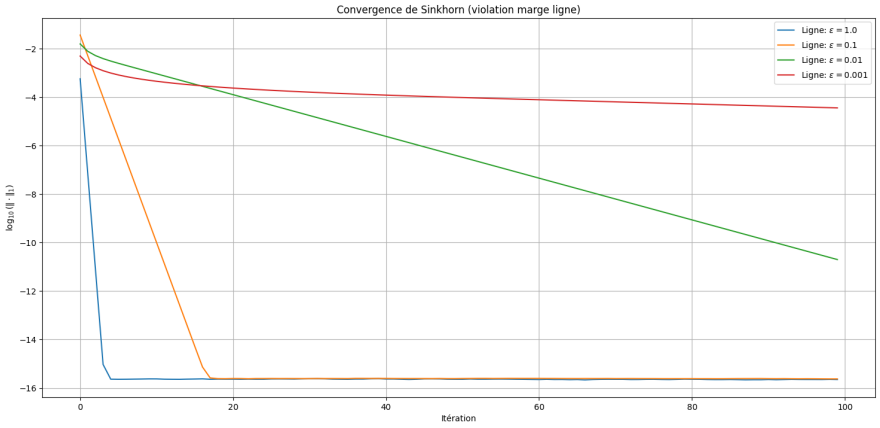


FIGURE 3 – Erreur de convergence suivant ε, avec des distributions de masse uniformes identiques

On observe en effet graphiquement des courbes présentant une tendance linéaire tant que la limite du zéro machine n'est pas atteinte.

Lorsque nous représentons les différents coefficients directeurs de chacune des droites, induites par les valeurs de ε, qui apparaissent avant le zéro machine, nous avons le graphe suivant :

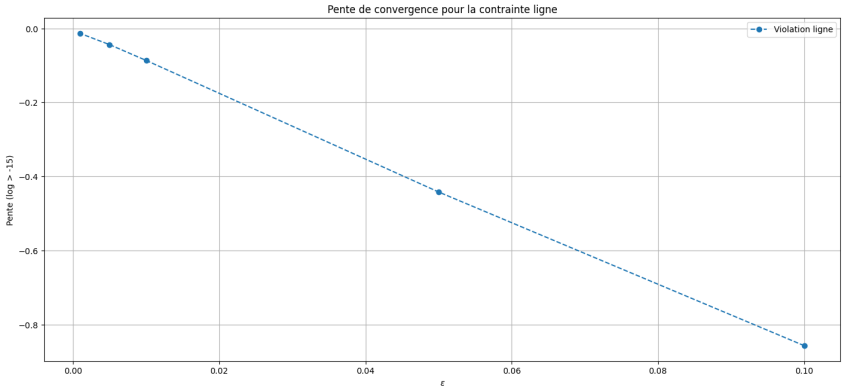


FIGURE 4 – Représentation graphique de $A(\varepsilon)$.

Des méthodes de régression linéaire avec la fonction `linregress`, disponible dans le module `stats` de la bibliothèque `scipy`, montrent que cette droite a une pente d'environ 8,6. Ainsi, on a $A(\varepsilon) \approx 8{,}6\,\varepsilon$, ce qui montre, pour $\varepsilon \sim 10^{-3}$, une décroissance extrêmement lente de l'erreur et nécessitant un nombre exorbitant d'itérations pour atteindre une erreur marginale faible.

2.2.3 Une visualisation de la trajectoire induite par un plan de transport

Jusqu'ici, nous avons étudié le transfert de masse entre deux distributions, sans décrire la trajectoire suivie. Cette section illustre l'évolution d'une distribution initiale au cours du transport.

Nous utilisons l'algorithme de Sinkhorn pour calculer un plan de transport et générer une série d'images montrant la répartition de la masse. À chaque instant $t \in [0, 1]$, les positions des masses sont interpolées géodésiquement par :

$$z_{i,j}^{(t)} = (1-t)x_i + ty_j,$$

où (x_i, y_j) sont des points de la grille source et cible pondérés par le plan de transport, définissant ainsi la distribution intermédiaire α_t. Cette dernière est donnée par :

$$\alpha_t \;=\; \sum_{i=1}^{n}\sum_{j=1}^{m} P_{ij}\,\delta_{z_{i,j}^{(t)}}.$$

Numériquement, à la date $t \in [0, 1]$ on cherche la masse totale en un point z_k de la grille de discrétisation. Ainsi on a :

$$a_t(k) = \sum_{i=1}^{n}\sum_{j=1}^{m} \mathbf{1}_{[z_k, z_{k+1}]}(z_{i,j}^{(t)})\,(1-\mu_{ij})\,P_{ij}, \qquad \mu_{ij} := \frac{z_{i,j}^{(t)} - z_k}{z_{k+1} - z_k}.$$

Le paramètre μ_{ij} mesure la distance relative à l'intérieur du bin et sert à répartir linéairement la masse entre les deux points successifs voisins z_k et z_{k+1}. Le calcul de $a_t(k+1)$ se fait de manière semblable mais on multiplie par μ_{ij} au lieu de $1-\mu_{ij}$.

Pour analyser cette dynamique, un plan de transport entre la distribution initiale α et α_t est recalculé à chaque instant, permettant de suivre l'évolution de la structure du transport. On présentera une généralisation de ce procédé avec plusieurs mesures initiales dans la section 2.4.

Pour illustrer cette méthode, nous reprenons l'exemple du transport entre deux mesures uniformes, toujours avec `max_iter` $= 1000$ et `tol` $= 10^{-9}$. Nous obtenons le résultat suivant :

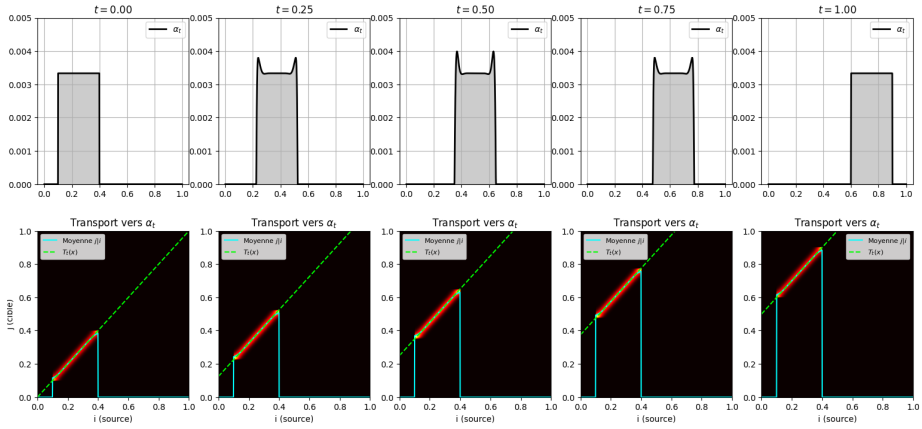


FIGURE 5 – Evolution du plan de transport entre deux mesures uniformes dans le temps, avec $\varepsilon = 0.001$

L'animation obtenue montre clairement l'évolution de la masse de α vers β, ainsi que l'adaptation progressive du plan de transport. Les contours sont lissés en raison de la régularisation entropique et la courbe de moyenne se superpose très bien pendant le déplacement avec la courbe de transport théorique qui est ici donnée par :

$$T_t(x) = (1-t)x + t(x+\tau)$$

Cette expérience montre concrètement l'action d'un plan de transport sur une masse initiale et illustre la qualité de l'approximation du transport optimal par

sa version entropiquement régularisée, surtout pour une régularisation importante.

2.3 Amélioration de l'algorithme de Sinkhorn par passage au log-domaine

Nous avons présenté dans la section 2.2 l'algorithme de Sinkhorn, qui permet de calculer des approximations numériques des plans de transport régularisés. Cependant, cet algorithme peut souffrir d'instabilités numériques lorsque le paramètre de régularisation ε est de l'ordre de 10^{-4}. Pour remédier à ce problème, il est possible de reformuler l'algorithme dans le domaine logarithmique, ce qui améliore sa stabilité numérique. Le passage dans ce domaine repose sur la propriété suivante.

Proposition 8 : *On a :*

$$L_C^{\varepsilon}(a,b) = \max_{\mathsf{f}\in\mathbb{R}^n,\mathsf{g}\in\mathbb{R}^m} \left(\langle \mathsf{f}, \mu\rangle + \langle \mathsf{g}, \nu\rangle + \varepsilon \left\langle e^{\mathsf{f}/\varepsilon}, K e^{\mathsf{g}/\varepsilon} \right\rangle \right) \tag{8}$$

Les valeurs optimales f et g sont d'ailleurs reliées aux vecteurs u et v de la proposition 7 par l'égalité :

$$(u,v) = \left(e^{\mathsf{f}/\varepsilon},\ e^{\mathsf{g}/\varepsilon} \right). \tag{9}$$

Une méthode simple pour résoudre (8) consiste à mettre à jour alternativement f et g afin d'annuler les gradients respectifs de l'objectif de (8) par rapport à ces variables. En effet, après quelques calculs élémentaires, en notant $Q(\mathsf{f},\mathsf{g})$ l'objectif de (8), on obtient :

$$\begin{cases} \nabla_{\mathsf{f}} Q(\mathsf{f},\mathsf{g}) = a - e^{\mathsf{f}/\varepsilon} \odot \left(K e^{\mathsf{g}/\varepsilon} \right), \\ \nabla_{\mathsf{g}} Q(\mathsf{f},\mathsf{g}) = b - e^{\mathsf{g}/\varepsilon} \odot \left(K^\top e^{\mathsf{f}/\varepsilon} \right) \end{cases}$$

Ainsi, cette méthode peut être mise en œuvre sous forme explicite en appliquant successivement les mises à jour suivantes :

$$\begin{cases} \mathsf{f}^{(l+1)} = \varepsilon \log(a) - \varepsilon \log\left(K e^{\mathsf{g}^{(l)}/\varepsilon} \right), \\ \mathsf{g}^{(l+1)} = \varepsilon \log(b) - \varepsilon \log\left(K^\top e^{\mathsf{f}^{(l+1)}/\varepsilon} \right) \end{cases} \tag{10}$$

C'est ainsi que se réalise le passage au log-domaine dans l'algorithme de Sinkhorn [5].

Cependant, dans ces itérations, les calculs logarithmiques en fin de ligne peuvent devenir instables lorsque ε est petit, en raison d'exponentielles de très grandes

5. Par passage au logarithme de (7) on obtient aussi (10) grâce à (9).

valeurs négatives. Cela provoque des sous-dépassements et des valeurs proches de zéro, rendant le logarithme numériquement instable. Ces calculs s'écrivent :

$$S_\varepsilon(z) := -\varepsilon \log \left(\sum_i e^{-z_i/\varepsilon} \right),$$

où z désigne un vecteur.

Pour éviter ces problèmes numériques, on utilise une technique de stabilisation appelée *log-sum-exp stabilisé*. En notant $\underline{z} := \min_i z_i$, on réécrit :

$$S_\varepsilon(z) = \underline{z} - \varepsilon \log \left(\sum_i e^{-(z_i - \underline{z})/\varepsilon} \right).$$

Cette reformulation est préférable car elle garantit qu'au moins l'un des termes de la somme (correspondant aux indices où $z_i = \underline{z}$) vaut $e^0 = 1$, ce qui empêche l'argument du logarithme d'être trop proche de zéro. Ainsi, même lorsque ε est très petit, cette écriture permet d'éviter les instabilités numériques. On choisit par ailleurs le critère d'erreur suivant :

$$\texttt{err} = \|\mathtt{f}^{(l+1)} - \mathtt{f}^{(l)}\|_1 + \|\mathtt{g}^{(l+1)} - \mathtt{g}^{(l)}\|_1 < \texttt{tol}$$

car ce critère, en log-domain, constitue un proxy fidèle de la violation des contraintes marginales et permet de mesurer la convergence globale de l'algorithme de manière stable.

Nous appliquons maintenant cette méthodologie au même exemple que dans la section 2.2.1. Nous pouvons à présent compléter les résultats de la la figure 2 en prenant $\varepsilon \in \{10^{-4}, 10^{-5}\}$. Le graphe qui suit a été obtenu avec `max_iter` $= 10^4$ et `tol` $= 10^{-9}$.

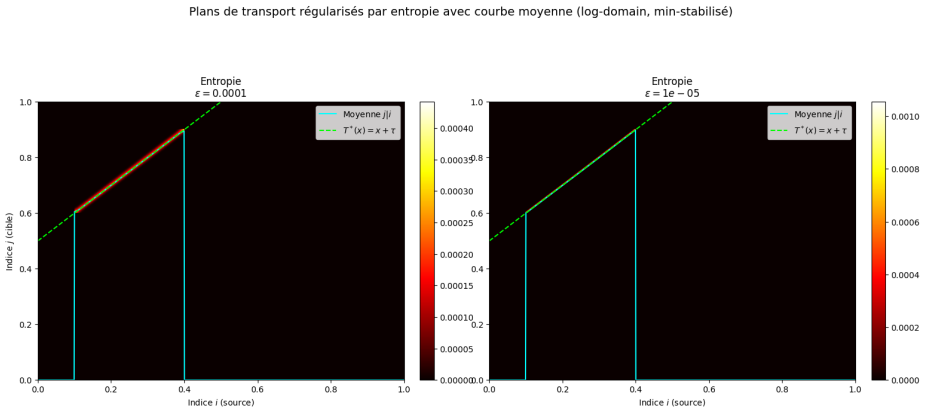


FIGURE 6 – Plans de transport entropiques en domaine logarithmique

On observe des résultats beaucoup plus précis à tel point qu'il y a une superposition quasi parfaite des différentes courbes du graphe.
Cependant, ce procédé nécessite beaucoup plus de temps de calcul et d'itérations, comme l'illustre le graphe suivant.

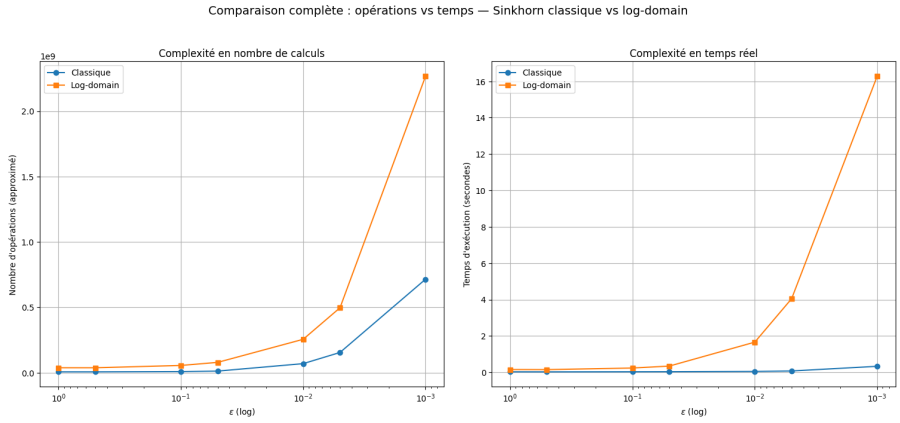


FIGURE 7 – Complexité calculatoire et temporelle des différentes méthodes de Sinkhorn

Une étude similaire à la section 2.2.2 montre que le coefficient directeur $A(\varepsilon) \approx 7,9\,\varepsilon$, ce qui illustre une meilleure précision de cette version de l'algorithme par rapport à la précédente.

2.4 Un cadre plus général, les barycentres entre plusieurs distributions

Dans cette section, nous considérons R mesures $(\alpha^1, \alpha^2, \ldots, \alpha^R) \in \mathcal{P}(\mathcal{X})^R$, pondérées respectivement par $(\lambda^1, \lambda^2, \ldots, \lambda^R) \in \mathbb{R}_+^R$ et telles que $\sum_{k=1}^{R} \lambda^k = 1$. L'objectif que nous allons nous fixer est de trouver le barycentre de cette famille de mesures. Pour cela, nous avons besoin de définir une métrique. Ceci se fera par l'intermédiaire de ce que l'on appelle la **distance de Wasserstein**.

Définition 9 : *Soit $(\mathcal{X}, d)$ un espace polonais muni de sa tribu borélienne. Soit $p \in [0, +\infty[$ et α, β deux mesures sur $\mathcal{X}$. La distance de Wasserstein d'ordre p entre α et β est définie par :*

$$W_p(\alpha, \beta) := \left(\inf_{\pi \in \mathcal{U}(\alpha,\beta)} \int_{\mathcal{X}\times\mathcal{X}} d(x,y)^p \, \mathrm{d}\pi(x,y) \right)^{1/p} \tag{11}$$

où $\mathcal{U}(\alpha, \beta)$ désigne l'ensemble des mesures de probabilités sur $\mathcal{X} \times \mathcal{X}$ dont les lois marginales sont α et β.

Dans ce qui va suivre, nous allons travailler avec $p = 2$ et $d(x, y) = \|x - y\|$. Nous définissons maintenant le barycentre relativement à cette métrique.

Définition 10 : *On définit le barycentre de notre distribution de mesure pondérée par :*

$$\bar{\alpha} := \arg \min_{\alpha \in \mathcal{P}(\mathcal{X})} \sum_{k=1}^{R} \lambda^k W_2^2(\alpha, \alpha^k) \tag{12}$$

Ce barycentre généralise la moyenne euclidienne à l'espace des mesures de distribution. De par le choix de p et d, on peut régulariser (11) par (6). Le problème régularisé devient donc :

$$\min_{a \in \Sigma_n} \sum_{k=1}^{R} \lambda^k \, L_C^{\varepsilon}(a, a^k) \tag{13}$$

Ceci revient alors à résoudre :

$$\bar{\mathbf{P}} := \arg \min_{\mathbf{P} \in \mathcal{U}(\boldsymbol{a})} \sum_{k=1}^{R} \lambda^k \, \mathrm{KL}(P^k | K) \tag{14}$$

où :

— $\boldsymbol{a} := \left(a^k\right)_{k=1}^{R} \in (\Sigma_n)^R$,

— $\mathbf{P} := \left(P^k\right)_{k=1}^{R}$,

— L'ensemble $\mathcal{U}(\boldsymbol{a})$ est défini par :

$$\begin{aligned} \mathcal{U}(\boldsymbol{a}) \coloneqq & \left\{ \mathbf{P} \in \left(\mathbb{R}_+^{n \times n}\right)^R \,\middle|\, \forall k \in \{1, \ldots, R\},\ P^k \mathbf{1}_n = a^k \right\} \\ & \cap \left\{ \mathbf{P} \in \left(\mathbb{R}_+^{n \times n}\right)^R \,\middle|\, \exists a \in \Sigma_n,\ \forall k \in \{1, \ldots, R\},\ (P^k)^\top \mathbf{1}_n = a \right\} \end{aligned}$$

— P^k est la matrice de transport optimisée entre α^k et α,

— $\mathrm{KL}(P^k \mid K) := \sum_{i=1}^{R} \sum_{j=1}^{R} P_{i,j}^k \left(\log\left(\frac{P_{i,j}^k}{K_{i,j}} \right) - 1 \right)$ est l'opérateur de Kullback-Leibler.

La méthode numérique, pour résoudre de manière approchée le problème, est basée sur la proposition suivante.

Proposition 11 : $\mathbf{P}$ satisfait, pour tout $1 \leq k \leq R$,

$$P^k = \mathrm{diag}\left(\frac{\bar{a}}{K\mathbf{1}_m} \right) K \quad \text{où} \quad \bar{a} := \prod_{k=1}^{R} \left(K\mathbf{1}_m\right)^{\lambda^k}$$

La solution peut être ainsi approchée par minimisation alternée, en alternant entre mise à jour des P^k et de $\bar{a}$ par l'intermédiaire de **l'algorithme de Sinkhorn**. Comme montré dans la section 2.2, on vérifie que les itérations, dans le

cas particulier du problème (14), donnent lieu à des itérés $\mathbf{P}^{(l)} := \left(P^{k,(l)}\right)_{k=1}^{R}$ satisfaisant, pour tout $1 \leq k \leq R$,

$$P^{k,(l)} = \mathrm{diag}(u^{k,(l)})\, K\, \mathrm{diag}(v^{k,(l)})$$

où $(u^{k,(l)}, v^{k,(l)}) \in {\mathbb{R}_+}^n \times {\mathbb{R}_+}^n$ sont deux vecteurs initialisés par $u^{k,(0)} = v^{k,(0)} = \mathbf{1}_n$ pour tout k, et calculés selon les itérations suivantes :

$$u^{k,(l+1)} := \frac{\bar{a}^{(l+1)}}{Kv^{k,(l)}} \quad \text{et} \quad v^{k,(l+1)} := \frac{a^k}{K^\top u^{k,(l+1)}}$$

où $\bar{a}^{(l)}$ désigne l'estimation du barycentre à l'itération l, calculée comme :

$$\bar{a}^{(l+1)} := \prod_{k=1}^{R} \left(u^{k,(l)} \odot (Kv^{k,(l)})\right)^{\lambda^k}$$

De manière semblable à la section 2.2 le critère d'arrêt est le suivant :

$$\texttt{err} = \sum_{k=1}^{R} \left(\|P_l^{k,(\varepsilon)} \mathbf{1}_n - a^k\|_1\right) < \texttt{tol}$$

Si l'on applique cela, par exemple, à trois distributions différentes, alors le barycentre obtenu présente une structure comme celle illustrée dans la figure suivante.

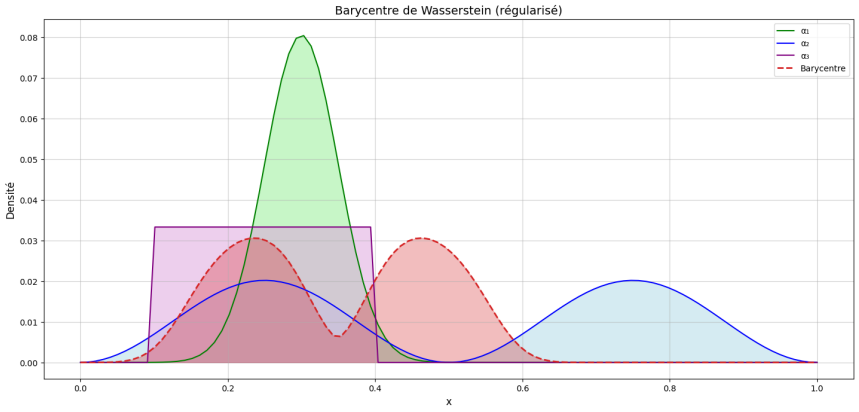


FIGURE 8 – Barycentre équipondéré entre une distribution gaussienne, uniforme et sinusoïdale

Notons que nous pouvons également adapter cet algorithme dans le *log domain*. En effet, d'après les résultats de la section 2.3, on a $u_k = e^{\mathrm{f}_k/\varepsilon}$ et $v_k = e^{\mathrm{g}_k/\varepsilon}$. On définit alors $q \in \mathbb{R}^n$ tel que $\bar{a} = e^{-q/\varepsilon}$. Ainsi, les itérations s'écrivent :

$$\begin{cases} \mathsf{f}_k^{(l+1)} = -q^{(l+1)} - \varepsilon \log\Big(Ke^{\mathsf{g}_k^{(l)}/\varepsilon}\Big), \\ \mathsf{g}_k^{(l+1)} = \varepsilon \log(a^k) - \varepsilon \log\Big(K^\top e^{\mathsf{f}_k^{(l+1)}/\varepsilon}\Big), \\ q^{(l+1)} = -\varepsilon \displaystyle\sum_{k=1}^{R} \lambda_k \log\Big(e^{\mathsf{f}_k^{(l)}} \odot Ke^{\mathsf{g}_k/\varepsilon}\Big) \\ \qquad\quad = \displaystyle\sum_{k=1}^{R} \lambda_k \left(-\mathsf{f}_k^{(l)} - \varepsilon \log\Big(Ke^{\mathsf{g}_k^{(l)}/\varepsilon}\Big)\right) \end{cases}$$

Avec bien sûr l'emploi de la méthode *log-sum-exp stabilisé* pour les calculs. Fidèlement à la section 2.3, le critère d'erreur est ici :

$$\texttt{err} = \sum_{k=1}^{R} (\|\mathsf{f}_k^{(l+1)} - \mathsf{f}_k^{(l)}\|_1 + \|\mathsf{g}_k^{(l+1)} - \mathsf{g}_k^{(l)}\|_1) < \texttt{tol}.$$

3 Application concrète du transport optimal

Cette section présente brièvement un domaine d'application du transport optimal : l'imagerie. Dans ce contexte, le transport de masse permet de modéliser des phénomènes de déplacement ou de transformation avec une grande précision géométrique.

3.1 Application à l'imagerie

Le transport optimal est largement utilisé en traitement d'image. Ici, nous montrons comment approcher numériquement des images intermédiaires reliant un ensemble d'images en niveaux de gris, vues comme des distributions de masse discrètes sur une grille $N \times N$. Les données sont représentées par des tenseurs. [6] Les images sont vectorisées en tenseurs 1D de taille N^2, les positions spatiales des pixels forment un tenseur $(N^2 \times 2)$, et les matrices de coût et plans de transport sont des tenseurs 2D de taille $N^2 \times N^2$. En exploitant la **séparabilité du noyau de Gibbs** associé au coût quadratique, on peut écrire le produit :

$$K.a = (K_1 \cdot a) \cdot K_1,$$

où K_1 est le noyau de Gibbs en 1D, ce qui accélère le calcul du produit $K.a$ dans l'algorithme de Sinkhorn. L'ensemble des images ainsi obtenues représente la transition continue entre les images sources selon le transport optimal régularisé. On obtient notamment le résultat suivant avec trois images, $N = 120$, $\varepsilon = 0.001$, `max_iter` $= 1000$ et `tol` $= 10^{-9}$.

6. Un *tenseur* est une structure de données multidimensionnelle qui généralise les scalaires, les vecteurs et les matrices. On peut imaginer cela comme un tableau à plusieurs dimensions.

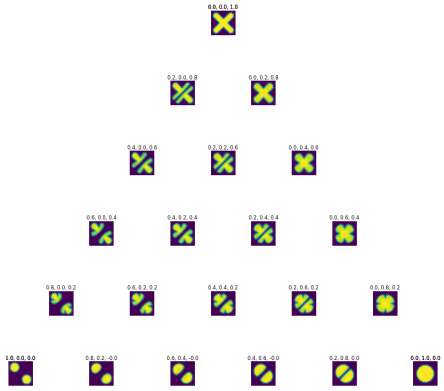


FIGURE 9 – Triangle barycentrique entre trois formes simples

Nous voyons sur cette figure une animation montrant les différentes transitions possibles entre les trois images insérées sur les trois côtés du triangle barycentrique.

Conclusion

Ce stage a permis d'explorer de manière approfondie la théorie et la résolution numérique du transport optimal, en reliant les formulations classiques de Monge et Kantorovich aux méthodes modernes de régularisation entropique. L'implémentation et l'analyse de l'algorithme de Sinkhorn, y compris dans sa version log-domaine, ont mis en évidence l'impact du paramètre ε sur la précision et la stabilité des calculs.
Les applications réalisées en imagerie ont illustré la flexibilité de ces méthodes et confirmé leur potentiel pour modéliser des phénomènes complexes. Ce travail a ainsi renforcé la compréhension des liens entre analyse théorique et outils numériques, tout en ouvrant des perspectives vers des applications plus larges et des optimisations à grande échelle.
Le lien entre le transport optimal et les équations aux dérivées partielles, qui constitue un domaine d'intérêt personnel, n'a malheureusement pas pu être traité dans le cadre de ce stage. Néanmoins, des références bibliographiques m'ont été transmises afin de me permettre d'approfondir ce sujet de manière autonome.

Annexes

Quelques preuves

Preuve du théorème 4 :

On considère deux distributions réelles α et β, et leurs fonctions de répartition respectives F_α et F_β. Si T^* est un transport optimal, alors, par la contrainte $T^*_\#\alpha = \beta$, on a, pour tout $x \in \mathbb{R}$:

$$F_\alpha(x) = F_\beta(T^*(x)).$$

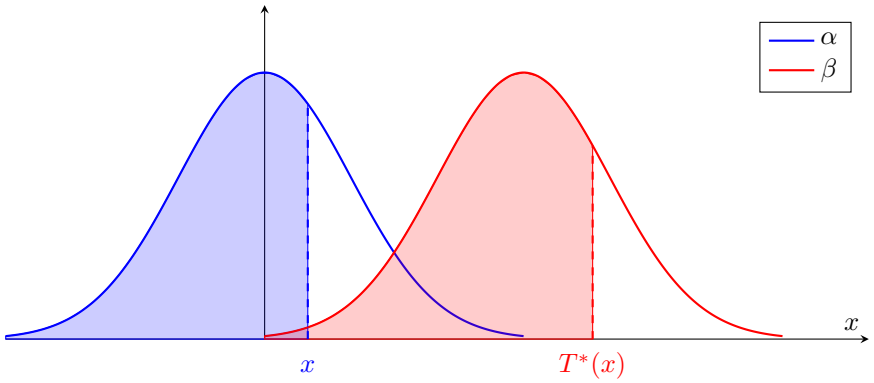


FIGURE 10 – Illustration de l'égalité $F_\alpha(x) = F_\beta(T^*(x))$ via les aires sous les densités.

On en déduit donc que, pour tout $x \in \mathbb{R}$,

$$T^*(x) = F_\beta^{-1}(F_\alpha(x)).$$

Réciproquement, si l'on se donne un transport T^* défini par

$$T^* = F_\beta^{-1} \circ F_\alpha,$$

alors, par la croissance des fonctions de répartition, on en déduit que T^* est aussi croissante. De plus on a que $T^*_\#\alpha = \beta$. En effet comme nous devons montrer une égalité entre deux mesures, nous raisonnons à l'aide de leurs fonctions de répartition. Il suffit de prouver que, pour tout $z \in \mathbb{R}$,

$$T^*_\#\alpha\big(]-\infty, z]\big) = \beta\big(]-\infty, z]\big).$$

On a

$$T^*_\#\alpha\big(]-\infty, z]\big) = \alpha\big(\{x \in \mathbb{R} : T^*(x) \leq z\}\big) = \alpha\big(\{x \in \mathbb{R} : F_\beta^{-1}(F_\alpha(x)) \leq z\}\big).$$

Comme F_β est strictement croissante et continue, on obtient l'équivalence

$$F_\beta^{-1}(F_\alpha(x)) \leq z \iff x \leq F_\alpha^{-1}(F_\beta(z)).$$

Ainsi

$$T_\#^*\alpha\big(]-\infty, z]\big) = \alpha\big(]-\infty, F_\alpha^{-1}(F_\beta(z))]\big) = F_\alpha\big(F_\alpha^{-1}(F_\beta(z))\big).$$

d'où

$$T_\#^*\alpha\big(]-\infty, z]\big) = F_\beta(z) = \beta\big(]-\infty, z]\big).$$

La contrainte $T_\#^*\alpha = \beta$ est donc vérifiée. Ainsi, T^* vérifie le théorème 3 et est donc optimal.
On conclut donc que, dans le cas réel,

$$T^* = F_\beta^{-1} \circ F_\alpha.$$

Ceci achève la preuve. □

Preuve de la proposition 6 :

On raisonne par caractérisation séquentielle. Soit $(\varepsilon_\ell)_{\ell\in\mathbb{N}}$, une suite telle que $\lim\limits_{\ell\to+\infty} \varepsilon_\ell = 0$. On note par P^ℓ, les solutions de (6) régularisé avec $\varepsilon_\ell > 0$.

L'ensemble $U(a, b)$ est un fermé borné. Ainsi, cet ensemble est compact. De ce fait, $(P^\ell)_{\ell\in\mathbb{N}}$ admet une suite extraite $(P^{\phi(\ell)})_{\ell\in\mathbb{N}}$ convergente et on note P^* sa limite. Aussi, $P^* \in U(a, b)$ car $U(a, b)$ est fermé.

On considère maintenant P une solution de (5). Ainsi, par optimalité de $P^{\phi(\ell)}$, on a :

$$C(P^{\phi(\ell)}) + \varepsilon_{\phi(\ell)} H(P^{\phi(\ell)}) \leq C(P) + \varepsilon_{\phi(\ell)} H(P).$$

Ainsi on a :

$$C(P^{\phi(\ell)}) - C(P) \leq \varepsilon_{\phi(\ell)}\big(H(P) - H(P^{\phi(\ell)})\big).$$

De plus, par optimalité de P, on a :

$$C(P) \leq C(P^{\phi(\ell)}).$$

D'où :

$$0 \leq C(P^{\phi(\ell)}) - C(P).$$

Donc on a finalement :

$$0 \leq C(P^{\phi(\ell)}) - C(P) \leq \varepsilon_{\phi(\ell)}\big(H(P) - H(P^{\phi(\ell)})\big). \tag{15}$$

Or H est continue, donc par passage à la limite quand $\ell \to +\infty$, on a $C(P) = C(P^*)$. Ainsi P^* est une solution optimale de (5).

Il nous faut maintenant prouver la convergence de $(P^\ell)_{\ell\in\mathbb{N}}$. Pour cela, on étudie le problème :

$$\min_{\substack{P\in U(a,b)\\ C(P)=L_C(a,b)}} H(P) \qquad (*)$$

Par stricte convexité de H, le problème admet une unique solution. Or, en divisant par $\varepsilon_{\phi(\ell)}$ dans (15) et quand $\ell \to +\infty$, on a $H(P^*) \leq H(P)$. Ainsi, P^* est une solution de (*). Donc P^* est unique, c'est donc l'unique valeur d'adhérence de $(P^\ell)_{\ell\in\mathbb{N}}$. Comme $U(a,b)$ est compact, on a :

$$\lim_{\ell\to+\infty} P^\ell = P^*.$$

Ceci conclut la preuve. □

Preuve de la proposition 7 :

On cherche à résoudre le problème d'optimisation :

$$L_C^\varepsilon(a,b) = \min_{P\in U(a,b)} \left(C(P) + \varepsilon H(P)\right)$$

et on rappelle que :

$$U(a,b) := \left\{P \in (\mathbb{R}_+)^{n\times m} \;\middle|\; P\mathbf{1}_m = a,\ P^T\mathbf{1}_n = b\right\}.$$

Nous savons déjà que la fonction objectif est strictement convexe. De plus, elle est différentiable et les contraintes sont affines. Il existe donc une unique solution optimale P^ε vérifiant les conditions KKT.

En écrivant $\mathcal{L}$ le Lagrangien et $\mathsf{f} \in \mathbb{R}^n$, $\mathsf{g} \in \mathbb{R}^m$ les multiplicateurs de Lagrange, on a :

$$\mathcal{L}(P,\mathsf{f},\mathsf{g}) = C(P) + \varepsilon H(P) - \langle \mathsf{f}, P\mathbf{1}_m - a\rangle - \langle \mathsf{g}, P^T\mathbf{1}_n - b\rangle,$$

soit, en développant :

$$\begin{aligned}\mathcal{L}(P,\mathsf{f},\mathsf{g}) = &\sum_{i=1}^n\sum_{j=1}^m C_{i,j}P_{i,j} + \varepsilon\sum_{i=1}^n\sum_{j=1}^m P_{i,j}\left(\log(P_{i,j})-1\right)\\ &- \sum_{i=1}^n \mathsf{f}_i\left(\sum_{j=1}^m P_{i,j} - a_i\right) - \sum_{j=1}^m \mathsf{g}_j\left(\sum_{i=1}^n P_{i,j} - b_j\right).\end{aligned}$$

P^ε vérifie donc, pour tout $(i,j) \in \{1,\dots,n\}\times\{1,\dots,m\}$:

$$\frac{\partial\mathcal{L}}{\partial P_{i,j}}(P^\varepsilon,\mathsf{f},\mathsf{g}) = 0,$$

d'où :

$$C_{i,j} + \varepsilon \log(P^{\varepsilon}_{i,j}) - \mathsf{f}_i - \mathsf{g}_j = 0.$$

On obtient donc :

$$P^{\varepsilon}_{i,j} = \exp\left(\frac{\mathsf{f}_i}{\varepsilon}\right) \exp\left(-\frac{C_{i,j}}{\varepsilon}\right) \exp\left(\frac{\mathsf{g}_j}{\varepsilon}\right).$$

On pose alors :

$$K_{i,j} := \exp\left(-\frac{C_{i,j}}{\varepsilon}\right), \quad u_i := \exp\left(\frac{\mathsf{f}_i}{\varepsilon}\right), \quad v_j := \exp\left(\frac{\mathsf{g}_j}{\varepsilon}\right).$$

On a donc :

$$P^{\varepsilon}_{i,j} = u_i K_{i,j} v_j.$$

Ceci permet de conclure. □

Preuve de la proposition 8 :

On reprend le résultat de la preuve qui précède. Nous avons prouvé que le plan de transport optimal était de la forme :

$$P^{\varepsilon}_{i,j} = \exp\left(\frac{\mathsf{f}_i}{\varepsilon}\right) \exp\left(-\frac{C_{i,j}}{\varepsilon}\right) \exp\left(\frac{\mathsf{g}_j}{\varepsilon}\right).$$

P^{ε} est la partie **primale** du **point selle lagrangien** et les vecteurs f et g constituent la partie **duale** . Ainsi, comme la fonction objectif est convexe et les contraintes sont affines, alors on a :

$$\min_{P \in \mathbb{R}_+^{n \times m}} \max_{\mathsf{f} \in \mathbb{R}^n, \mathsf{g} \in \mathbb{R}^m} \mathcal{L}(P, \mathsf{f}, \mathsf{g}) = L^{\varepsilon}_C(a, b)$$

Donc,

$$\min_{P \in \mathbb{R}_+^{n \times m}} \max_{\mathsf{f} \in \mathbb{R}^n, \mathsf{g} \in \mathbb{R}^m} \mathcal{L}(P, \mathsf{f}, \mathsf{g}) = \max_{\mathsf{f} \in \mathbb{R}^n, \mathsf{g} \in \mathbb{R}^m} \mathcal{L}(P^{\varepsilon}, \mathsf{f}, \mathsf{g}).$$

Or,

$$\begin{aligned}\mathcal{L}(P^{\varepsilon}, \mathsf{f}, \mathsf{g}) = &\sum_{i=1}^{n}\sum_{j=1}^{m} C_{i,j} P^{\varepsilon}_{i,j} + \varepsilon \sum_{i=1}^{n}\sum_{j=1}^{m} P^{\varepsilon}_{i,j} \left(\log(P^{\varepsilon}_{i,j}) - 1\right) \\ &- \sum_{i=1}^{n} \mathsf{f}_i \left(\sum_{j=1}^{m} P^{\varepsilon}_{i,j} - a_i\right) - \sum_{j=1}^{m} \mathsf{g}_j \left(\sum_{i=1}^{n} P^{\varepsilon}_{i,j} - b_j\right).\end{aligned}$$

Or, $\log(P^{\varepsilon}_{i,j}) = \frac{\mathsf{f}_i - C_{i,j} + \mathsf{g}_j}{\varepsilon}$. Donc

$$\mathcal{L}(P^\varepsilon, \mathsf{f}, \mathsf{g}) = \sum_{i=1}^{n}\sum_{j=1}^{m} C_{i,j}P^\varepsilon_{i,j} + \sum_{i=1}^{n}\sum_{j=1}^{m} P^\varepsilon_{i,j}\left(\mathsf{f}_i - C_{i,j} + \mathsf{g}_j - \varepsilon\right)$$
$$- \sum_{i=1}^{n} \mathsf{f}_i \left(\sum_{j=1}^{m} P^\varepsilon_{i,j} - a_i\right) - \sum_{j=1}^{m} \mathsf{g}_j \left(\sum_{i=1}^{n} P^\varepsilon_{i,j} - b_j\right).$$

Ceci se simplifie immédiatement en donnant :

$$\mathcal{L}(P^\varepsilon, \mathsf{f}, \mathsf{g}) = -\varepsilon \sum_{i=1}^{n}\sum_{j=1}^{m} P^\varepsilon_{i,j} + \sum_{i=1}^{n} \mathsf{f}_i a_i + \sum_{j=1}^{m} \mathsf{g}_j b_j.$$

C'est-à-dire, en passant en écriture vectorielle,

$$\mathcal{L}(P^\varepsilon, \mathsf{f}, \mathsf{g}) = \langle \mathsf{f}, a\rangle + \langle \mathsf{g}, b\rangle - \varepsilon \left\langle e^{\mathsf{f}/\varepsilon},\ K e^{\mathsf{g}/\varepsilon} \right\rangle.$$

Finalement, on a :

$$L^\varepsilon_C(a,b) = \max_{\mathsf{f}\in\mathbb{R}^n, \mathsf{g}\in\mathbb{R}^m} \mathcal{L}(P^\varepsilon, \mathsf{f}, \mathsf{g}) = \max_{\mathsf{f}\in\mathbb{R}^n, \mathsf{g}\in\mathbb{R}^m} \langle \mathsf{f}, a\rangle + \langle \mathsf{g}, b\rangle - \varepsilon \left\langle e^{\mathsf{f}/\varepsilon},\ K e^{\mathsf{g}/\varepsilon} \right\rangle.$$

La preuve est achevée. □

Codes Python

Les codes complets sont disponibles sur ce notebook Google Colab.

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import interp1d

# ------------------------------------------------------------
## Fonction : sinkhorn
# Implémente l'algorithme de Sinkhorn pour le transport optimal
# régularisé par entropie.
# ------------------------------------------------------------
# Paramètres :
# - a : vecteur de probabilité source (taille n, somme = 1)
# - b : vecteur de probabilité cible (taille m, somme = 1)
# - C : matrice de coûts (n × m), ici distance au carré
# - epsilon : coefficient de régularisation entropique
# - max_iter : nombre maximal d'itérations (par défaut 1000)
# - tol : tolérance sur l'erreur marginale pour arrêt anticipé
# ------------------------------------------------------------
# Retour :
# - P : matrice de transport (n × m) qui approxime la solution
# ------------------------------------------------------------
def sinkhorn(a, b, C, epsilon, max_iter=1000, tol=1e-9):
    n, m = C.shape
    K = np.exp(-C / epsilon)          # Noyau de Gibbs (matrice K)
    u = np.ones(n)                    # Potentiel multiplicatif u
    v = np.ones(m)                    # Potentiel multiplicatif v
    for _ in range(max_iter):
        u = a / np.dot(K, v)          # Mise à jour de u
        v = b / np.dot(K.T, u)        # Mise à jour de v
        P = np.dot(np.diag(u), np.dot(K, np.diag(v)))  # Plan de
            transport
        err = np.linalg.norm(P.sum(axis=1) - a, 1)     # Écart
            marginal source
        if err < tol:                 # Critère d'arrêt
            break
    return P
```

Listing 1 : Implémentation de l'algorithme de Sinkhorn en Python

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import interp1d

# ------------------------------------------------------------
## Fonction : sinkhorn_barycenter
# Calcule le barycentre de Wasserstein régularisé pour un ensemble
# de mesures discrètes à l'aide de l'algorithme de Sinkhorn.
# ------------------------------------------------------------
# Paramètres :
# - measures : liste de vecteurs de probabilités à barycentrer
# - lambda_ : poids associés à chaque mesure (somme = 1)
# - C : matrice des coûts quadratiques C[i, j] = (x_i - y_j)^2
# - epsilon : régularisation entropique
# - max_iter : nombre maximal d'itérations
# - tol : tolérance sur l'erreur marginale pour arrêt anticipé
# ------------------------------------------------------------
# Retour :
# - a_bar : vecteur de probabilités représentant le barycentre
# ------------------------------------------------------------
def sinkhorn_barycenter(measures, lambda_, C, epsilon=0.001,
    max_iter=4000, tol=1e-9):
    n = C.shape[0]              # Taille de la grille
    R = len(measures)                    # Nombre de mesures
    K = np.exp(-C / epsilon)    # Noyau de Gibbs

    # Empilement des mesures en colonnes et ajout d'une petite
        constante pour stabilité
    A = np.stack(measures, axis=1) + 1e-16

    # Initialisation des variables duales u et v (facteurs de
        scaling)
    u = np.ones((n, R))
    v = np.ones((n, R))

    # Initialisation du barycentre uniforme
    a_bar = np.ones(n) / n

    # Boucle principale de l'algorithme
    for _ in range(max_iter):

        # Étape 1 : calcul du barycentre comme moyenne géométrique
            pondérée
        log_a_bar = np.zeros(n)
        for k in range(R):
            uKv = u[:, k] * (K @ v[:, k])
            log_a_bar += lambda_[k] * np.log(uKv)
        a_bar = np.exp(log_a_bar)

        # Étape 2 : mise à jour des variables u (projection sur les
             secondes marginales)
        for k in range(R):
            Kv = K @ v[:, k]
            u[:, k] = a_bar / Kv

        # Étape 3 : mise à jour des variables v (projection sur les
             premières marginales)
```

```
        for k in range(R):
            Ku = K @ u[:, k]
            v[:, k] = A[:, k] / Ku

        # Calcul de l'erreur marginale totale
        err = 0
        for k in range(R):
            Pk = np.dot(np.diag(u[:, k]), np.dot(K, np.diag(v[:, k
                ])))
            err += np.linalg.norm(Pk.sum(axis=1) - A[:, k], 1)

        # Critère d'arrêt si erreur inférieure à tol
        if err < tol:
            break

    return a_bar
```

Listing 2 : Implémentation de l'algorithme de Sinkhorn pour les calculs de barycentre

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import interp1d
from scipy.special import logsumexp

# ------------------------------------------------------------
## Fonction : min_stabilized_logsumexp
# ------------------------------------------------------------
# Objectif :
#   Calculer l'opérateur de type :
#       -ε * log( sum_j exp( -Z / ε ) )
#   de façon numériquement stable.
# Méthode :
#   On extrait d'abord min(Z) pour éviter les
#   problèmes de dépassement ou d'annulation numérique dans l'
    exponentielle.
#
# Paramètres :
#   - Z : tableau des coûts ajustés (peut être 2D ou plus)
#   - epsilon : paramètre de régularisation entropique
#   - axis : axe de réduction pour la somme (par défaut 1)
#
# Retour :
#   - Tableau des valeurs stabilisées de -ε log-sum-exp
# ------------------------------------------------------------
def min_stabilized_logsumexp(Z, epsilon, axis=1):
    min_Z = np.min(Z, axis=axis, keepdims=True)  # Minimum par
        ligne/colonne
    stabilized = np.exp(-(Z - min_Z) / epsilon)  # Stabilisation
        avant exponentielle
    log_sum = logsumexp(-(Z - min_Z) / epsilon, axis=axis, keepdims
        =True)  # Somme en log
    return min_Z - epsilon * log_sum

# ------------------------------------------------------------
## Fonction : sinkhorn_log_domain
# ------------------------------------------------------------
# Objectif :
#   Implémenter l'algorithme de Sinkhorn en domaine logarithmique,
#   avec stabilisation par min, afin de traiter des valeurs très
    petites
#   d'epsilon (jusqu'à 1e-5 ou 1e-6) sans sous-flux numérique.
#
# Paramètres :
#   - a : distribution de probabilité source (taille n)
#   - b : distribution de probabilité cible (taille m)
#   - C : matrice de coût (n × m)
#   - epsilon : paramètre de régularisation entropique
#   - max_iter : nombre maximal d'itérations
#   - tol : tolérance pour l'arrêt anticipé sur convergence
#
# Retour :
#   - P : plan de transport optimal régularisé (n × m)
# ------------------------------------------------------------
def sinkhorn_log_domain(a, b, C, epsilon, max_iter=1000, tol=1e-9):
    n, m = C.shape
    f = np.zeros(n)  # Potentiel source (log)
```

```
    g = np.zeros(m)  # Potentiel cible (log)

    # Transformation en log pour éviter les divisions par zéro
    log_a = np.log(a + 1e-300)
    log_b = np.log(b + 1e-300)

    for _ in range(max_iter):
        f_prev = f.copy()
        g_prev = g.copy()

        # Mise à jour de f en domaine log
        Z_f = C - g[None, :]
        f = epsilon * log_a + min_stabilized_logsumexp(Z_f, epsilon
            , axis=1).flatten()

        # Mise à jour de g en domaine log
        Z_g = C.T - f[None, :]
        g = epsilon * log_b + min_stabilized_logsumexp(Z_g, epsilon
            , axis=1).flatten()

        # Critère d'arrêt : convergence des potentiels
        err = np.linalg.norm(f - f_prev) + np.linalg.norm(g -
            g_prev)
        if err < tol:
            break

        # Critère d'arrêt : convergence des potentiels
        #delta_f = np.max(np.abs(f - f_prev))
        #delta_g = np.max(np.abs(g - g_prev))
        #if max(delta_f, delta_g) < tol:
            #break

    # Reconstruction du plan P à partir des potentiels
    log_P = -(C - f[:, None] - g[None, :]) / epsilon
    P = np.exp(log_P)
    return P
```

Listing 3 : Implémentation de l'algorithme de Sinkhorn en log domaine

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import interp1d
from scipy.special import logsumexp

# ------------------------------------------------------------
## Fonction : min_stabilized_logsumexp
# Calcule le log-sum-exp stabilisé numériquement pour un vecteur ou une matrice
# Cette fonction est utilisée pour éviter les overflow/underflow lors
# des calculs log-domain dans l'algorithme de Sinkhorn.
# ------------------------------------------------------------
# Paramètres :
# - Z : matrice ou vecteur d'entrée
# - epsilon : régularisation entropique
# - axis : axe le long duquel effectuer le log-sum-exp
# ------------------------------------------------------------
# Retour :
# - log-sum-exp stabilisé
# ------------------------------------------------------------
def min_stabilized_logsumexp(Z, epsilon, axis=1):
    min_Z = np.min(Z, axis=axis, keepdims=True)                # Décalage pour la stabilité numérique
    log_sum = logsumexp(-(Z - min_Z) / epsilon, axis=axis, keepdims=True)
    return min_Z - epsilon * log_sum

# ------------------------------------------------------------
## Fonction : sinkhorn_barycenter_log
# Calcule le barycentre de Wasserstein régularisé en domaine logarithmique
# pour un ensemble de mesures discrètes à l'aide de l'algorithme de Sinkhorn.
# ------------------------------------------------------------
# Paramètres :
# - measures : liste de vecteurs de probabilités à barycentrer
# - lambdas : poids associés à chaque mesure (somme = 1)
# - C : matrice des coûts quadratiques C[i, j] = (x_i - y_j)^2
# - epsilon : régularisation entropique
# - max_iter : nombre maximal d'itérations
# - tol : tolérance sur les changements de variables duales pour arrêt anticipé
# ------------------------------------------------------------
# Retour :
# - a_bar : vecteur de probabilités représentant le barycentre
# ------------------------------------------------------------
def sinkhorn_barycenter_log(measures, lambdas, C, epsilon=0.001, max_iter=4000, tol=1e-9):
    n = C.shape[0]                                    # Taille de la grille
    R = len(measures)                                 # Nombre de mesures

    log_a = [np.log(measure + 1e-300) for measure in measures]  # Logarithme des mesures pour stabilité
    f = [np.zeros(n) for _ in range(R)]               # Potentiels duals
```

```
        f_k
    g = [np.zeros(n) for _ in range(R)]          # Potentiels duals
        g_k
    q = np.zeros(n)                               # Potentiel
        barycentrique
    a_bar = np.exp(-q / epsilon)                  # Barycentre
        initial uniforme

    for it in range(max_iter):
        f_prev = [fk.copy() for fk in f]          # Sauvegarde pour
            vérification de convergence
        g_prev = [gk.copy() for gk in g]

        # Calcul du barycentre : mise à jour du potentiel q
        q_new = np.zeros(n)
        for k in range(R):
            Z = C - g[k][None, :]                                  # Dé
                placement par g_k
            q_new += lambdas[k] * ( -f[k] +
                min_stabilized_logsumexp(Z, epsilon, axis=1).
                flatten() )
        q = q_new
        a_bar = np.exp(-q / epsilon)
                                              # Mise à jour du
            barycentre

        # Mise à jour des potentiels f_k
        for k in range(R):
            Z = C - g[k][None, :]
            f[k] = -q + min_stabilized_logsumexp(Z, epsilon, axis
                =1).flatten()

        # Mise à jour des potentiels g_k
        for k in range(R):
            Z = C.T - f[k][None, :]
            g[k] = epsilon * log_a[k] + min_stabilized_logsumexp(Z,
                 epsilon, axis=1).flatten()

        # Vérification de la convergence
        max_delta = 0
        for k in range(R):
            delta_f = np.max(np.abs(f[k] - f_prev[k]))
            delta_g = np.max(np.abs(g[k] - g_prev[k]))
            max_delta = max(delta_f, delta_g)

        if max_delta < tol:
            break

    return a_bar
```

Listing 4 : Implémentation de l'algorithme de Sinkhorn pour les calculs de barycentre en log domaine

Références

[1] Gabriel Peyré, Marco Cuturi. *Computational Optimal Transport*. Foundations and Trends in Machine Learning, vol. 11, n° 5-6, 2019, pp. 355–607. Disponible sur : `https://arxiv.org/abs/1803.00567`

[2] Léo Germain. *Transport Optimal Régularisé : méthodes de résolution et applications*. Thèse de doctorat en mathématiques appliquées, École Normale Supérieure Paris-Saclay, 2020. Disponible sur : `https://theses.hal.science/tel-03152994`

[3] Gabriel Peyré. *Numerical Tours of Data Science — Entropic Optimal Transport*. Notebook Python, 2020. Disponible sur : `https://nbviewer.org/github/gpeyre/numerical-tours/blob/master/python/optimaltransp_5_entropic.ipynb`

Résumé

Zakaria OTMANE
4ème année INSA Rennes
Grupo de Física Matemática, IST Lisbonne

Introduction théorique et numérique au transport optimal

Ce travail s'inscrit dans le cadre d'un stage de recherche en mathématiques appliquées, réalisé au sein du Grupo de Física Matemática de l'Instituto Superior Técnico de Lisbonne. L'objectif principal était d'étudier la théorie du transport optimal et ses méthodes de résolution numérique, en particulier l'algorithme de Sinkhorn pour le transport entropiquement régularisé. Après une présentation des formulations de Monge et de Kantorovich ainsi que de résultats fondamentaux (dont le théorème de Brenier), l'étude a porté sur la discrétisation numérique du problème, la régularisation entropique, l'analyse des erreurs de convergence et l'amélioration par passage au log-domaine. L'extension aux barycentres de Wasserstein a également été explorée. Les méthodes implémentées en Python ont permis de confirmer la validité théorique et d'illustrer des applications concrètes en traitement d'images, notamment la génération d'interpolations barycentriques entre plusieurs formes. Ce travail a permis d'approfondir le lien entre théorie mathématique et calcul scientifique, tout en ouvrant des perspectives vers des applications plus larges en modélisation et en imagerie numérique.

Mots clés : transport optimal, régularisation entropique, algorithme de Sinkhorn, log-domaine, barycentres de Wasserstein, traitement d'images.

Abstract

Zakaria OTMANE
4th year INSA Rennes
Grupo de Física Matemática, IST Lisbon

Theoretical and Numerical Introduction to Optimal Transport

This work was carried out during a research internship in applied mathematics at the Grupo de Física Matemática of Instituto Superior Técnico in Lisbon. The main objective was to study the theory of optimal transport and its numerical resolution methods, with a focus on the Sinkhorn algorithm for entropically regularized transport. After presenting Monge's and Kantorovich's formulations as well as fundamental results (including Brenier's theorem), the study focused on numerical discretization, entropic regularization, convergence error analysis, and stabilization through the log-domain approach. The extension to Wasserstein barycenters was also explored. Python implementations validated the theoretical framework and illustrated practical applications in image processing, in particular the generation of barycentric interpolations between several shapes. This work deepened the connection between mathematical theory and scientific computing, while opening perspectives for broader applications in modeling and digital imaging.

Keywords : optimal transport, entropic regularization, Sinkhorn algorithm, log-domain, Wasserstein barycenters, image processing.