AMO - Advanced Modeling and Optimization, Volume 12, Number 3, 2010

Optrans: A Parallel Software Library for Optimal Transport

Said Kerrache and Yasushi Nakauchi
{kerrache,nakauchi}@hri.iit.tsukuba.ac.jp
Graduate School of Systems and Information Engineering
University of Tsukuba
1-1-1 Tennodai, Tsukuba, Ibaraki 305-8577 Japan

Abstract

In recent years, optimal transport has become a highly active and wide area of research, thanks to the discovery of a number of important theoretical results and the development of an array of new applications in various fields ranging from cosmology, geophysics, oceanography, meteorology and fluid mechanics to optics, image processing and pattern recognition. Despite this ample field of applications, there is a serious lack of numerical optimal transport softwares available to the research and academic community. To remedy to this shortage, this paper introduces Optrans, a parallel library for solving time-dependent optimal transport problems in free and convexly constrained forms. Optrans is designed following an object oriented approach and exploits the capabilities of C++, the implementation language, to offer an easy-to-use programming interface and ensure easy-extendability. The library uses MPI for communication and synchronization, allowing it to run on a variety of architectures. In addition to the software related aspects of the library, a number of implemented numerical techniques are presented, for instance, solving transport problems on domains with null Neumann boundary conditions and using a smoothing term to transport densities that approach zero. Numerical experiments are presented to demonstrate the capabilities of the library and the implemented techniques.

Key words: Optimal transport, Monge-Kantorovich, Parallel programming, MPI, Software library.

1. Introduction

The optimal transport problem was first introduced more than 200 years ago by the French mathematician Monge, who studied the problem of transporting soil materials from mining sites to construction sites with a minimum cost. Monge assumed the cost to be proportional to the distance traveled and the mass transported. Later on, this problem came to be known as the optimal transport with the Euclidean distance. In the first half of the twentieth century, Kantorovich, a Russian mathematician and economist, made important contributions to the field by introducing a weak version of optimal transport and formulating the problem as a linear program [Kantorovitch, 1942]. Besides its application in economics, optimal transport has also been widely used in probability theory [Rachev and Ruschendorf, 1998]. In the last two decades, research on the subject has undergone a rapid expansion, and a wide range of new areas of application has been discovered. This includes fluid dynamics [Jordan et al., 1998,Carrillo et al., 2007], geophysics, oceanography and meteorology [Cullen, 2006,Cullen and Maroofi, 2003], cosmology [Frisch et al., 2002,Brenier et al., 2003], image processing and computer vision [Gangbo and McCann, 2000,Haker et al., 2004,Museyko et al., 2009] to cite a few.

There are two formulations of the optimal transport problem. In the classic, or time-independent formulation, a map from the space that contains the initial density to the space that contains the final density is sought, which minimizes the transport cost. In the time-dependent version, the two densities are assumed to lie on the same space, and the goal is to find a time-continuous transport plan that minimizes the transport cost. The time-dependent formulation offers a more complete description of the transport process, since it gives a time-continuous interpolation of the density. Optimal transport with the squared Euclidean distance as cost is by far the most theoretically understood and practically used instance of the optimal transport problem [Villani, 2009]. Its time-dependent version was discovered to admit a computational fluid dynamics formulation [Benamou and Brenier, 2000], which allowed the development of an iterative numerical scheme to solve the time-dependent transport problem with the squared Euclidean distance as cost on closed convex subsets of $\mathbb{R}^d$. The transport problem is shown equivalent to finding a flow of minimum kinetic energy that transports the initial density to the final one. In [Kerrache and Nakauchi, 2010], this problem is solved with the constraint that the interpolating density and the momentum of the flow belong to a closed convex set.

This paper introduces Optrans, a parallel library for solving the time-dependent optimal transport problem. The library implements the algorithm proposed in [Benamou and Brenier, 2000] for free problems and the family of algorithms proposed in [Kerrache and Nakauchi, 2010] for solving constrained problems. The time-dependent problem is computationally complex but lends itself to parallelism, which motivates the choice of a parallel implementation. To the best knowledge of the authors, there are no publicly available softwares for solving the time-dependent optimal transport problem, neither serial nor parallel.

This paper is organized as follows. Section 2 gives an overview of the algorithms implemented in Optrans and presents a number of numerical techniques used in their implementation. Section 3 details the internal architecture and the programming interface of the library. Section 4 presents a series of numerical experiments that demonstrate the working of Optrans and the implemented techniques. Finally, Section 5 concludes the paper and gives some future research directions.

2. Algorithms and Methods

In [Benamou and Brenier, 2000], the problem of optimal mass transport in a closed convex subset D of $\mathbb{R}^d$ with the squared Euclidean distance as cost is recast as an optimal control problem of a potential flow [Cohen and Kundu, 2004]. The approach consists in computing a flow that moves the initial density to the final one, while minimizing the kinetic energy. More precisely, the problem is formulated as

$$\inf_{\rho,m} \int_0^1 \int_D \frac{|m(t,x)|^2}{2\rho(t,x)} dxdt, \tag{1}$$

$$\text{s.t. } \partial_t \rho + \nabla \cdot m = 0, \quad \rho(0,\cdot) = \rho_0, \quad \rho(1,\cdot) = \rho_1, \tag{2}$$

where $\rho(t,x)$ is the density, m is the momentum of the flow, $\rho_0(x)$ and $\rho_1(x)$ are two bounded positive density functions defined on D, such as:

$$\int_D \rho_0(x)\, dx = \int_D \rho_1(x)\, dx = 1$$

This problem is then transformed to the following saddle point problem:

$$\inf_{\phi,q} \sup_{\mu} \mathcal{L}(\phi, q, \mu) = F(q) + G(\phi) + \langle \mu, \nabla\phi - q \rangle, \tag{3}$$

where $\mu = (\rho, m)$, $G(\phi) = \int_D \phi(0,x)\rho_0(x) - \phi(1,x)\rho_1(x)\, dx$, F is defined by

$$F(q) = \begin{cases} 0 & \text{if } q \in K, \\ +\infty & \text{otherwise,} \end{cases}$$

with

$$K = \left\{ (a,b) : \mathbb{R} \times \mathbb{R}^d \to \mathbb{R} \times \mathbb{R}^d, a + \frac{|b|^2}{2} \leq 0 \text{ pointwise} \right\},$$

and $\langle \cdot, \cdot \rangle$ is the inner product defined by

$$\langle u, v \rangle = \int_0^1 \int_D u \cdot v.$$

The authors then present an algorithm to compute the optimal flow by iteratively updating ϕ, q and μ. The computation of ϕ consists in solving a Poisson equation, q is obtained by solving a pointwise optimization problem, whereas μ is updated using a gradient-type rule.

In [Kerrache and Nakauchi, 2010], Problem (1) is solved under the additional constraint that $\mu = (\rho, m) \in U$, where U is a closed convex set. In the algorithms proposed to solve the constrained version, μ is obtained as the solution to a quadratic optimization problem having U as the constraint set.

Optrans implements the algorithm proposed in [Benamou and Brenier, 2000] and the set of algorithms proposed in [Kerrache and Nakauchi, 2010]. In the remainder of this section, a number of issues and techniques related to the implementation of these algorithms are presented.

2.1. *Computing q*

The optimization required to compute q can be performed numerically by using a general nonlinear optimization algorithm. However, an analytic solution can substantially reduce computation time and would therefore be preferable, especially for large problems. The objective function and the constraint of the problem are both convex and differentiable, which implies that the KKT (Karush-Kuhn-Tucker) conditions are necessary and sufficient to characterize the optimum. Remember that, at step n, q is obtained as the solution to (see [Benamou and Brenier, 2000] for details):

$$\inf_{q \in K} \left\langle \nabla \phi^n + \frac{\mu^n}{r} - q, \nabla \phi^n + \frac{\mu^n}{r} - q \right\rangle.$$

This problem can be solved pointwise. Let:

$$p(t,x) = (\alpha(t,x), \beta(t,x)) = \nabla \phi^n (t,x) + \frac{\mu^n (t,x)}{r}.$$

Then, the new value of $q(t,x) = (a(t,x), b(t,x))$ can be obtained by solving in (a, b)

$$\inf \left\{ (a-\alpha)^2 + |b-\beta|^2, \quad a + \frac{|b|^2}{2} \leq 0 \right\},$$

where the explicit dependency on x and t is dropped since the problem is to be solved pointwise. The Lagrangian for this problem is:

$$\mathcal{L}_q (a, b_i, \lambda) = (a-\alpha)^2 + |b-\beta|^2 - \lambda \left(a + \frac{|b|^2}{2} \right)$$

The KKT conditions for this problem are then:

$$\begin{cases} 2(a-\alpha) - \lambda = 0 \\ 2(b_i - \beta_i) - \lambda b_i = 0 \\ \lambda \left(a + \frac{|b|^2}{2} \right) = 0 \\ a + \frac{|b|^2}{2} \leq 0 \\ \lambda \leq 0 \end{cases} \tag{4}$$

Hence, λ must satisfy the following third degree equation:

$$\lambda^3 + (2\alpha - 4)\lambda^2 + (4 - 8\alpha)\lambda + 8\alpha + 4|\beta|^2 = 0$$

Although this equation can be solved analytically, such a method suffers from numerical instability. Dedicated and robust numerical methods for finding polynomial roots are better suited in this case. To this end, Optrans uses the Jenkins-Traub algorithm [Jenkins and Traub, 1970]. Once the roots of the polynomial are found, determining the optimal point is a matter of a straightforward computation.

2.2. *Handling null Neumann boundary conditions*

The algorithm presented in [Benamou and Brenier, 2000] is stated for domains with periodic boundary conditions. However, the method can be adapted to domains with null Neumann boundary conditions with a minimum change. In deed, a close inspection of the KKT optimality conditions for computing q (see Eq. (4)) reveals that the space components of q and $\nabla\phi + \mu/r$ must be either parallel or simultaneously null. From the update rule of μ, it can be seen that if $\nabla\phi^n$, q^n and μ^n have null normal component in the space domain, then so does μ^{n+1}. Therefore, if the initial value μ^0 has null normal component in the space domain and ϕ^n is obtained by solving the Poisson PDE with null Neumann boundary conditions in space, then all the terms of the sequence μ^n, and consequently its limit, satisfy the same condition.

2.3. *Smoothing term*

The algorithm proposed by [Benamou and Brenier, 2000] is an application of a more general numerical scheme, a description of which can be found in [Fortin and Glowinski, 1983] and [Glowinski and Tallec, 1989]. Theoretically, the optimal transport problem is not coercive as required, but this can be dealt with by adding to the Lagrangian $\mathcal{L}$ (see Eq. (3)) a small perturbation term: $H\,|q|^2$, where H is a positive parameter. However, as observed in [Benamou and Brenier, 2000], this perturbation is unnecessary in practice. Nonetheless, the additional term, $H\,|q|^2$, can be be useful to smooth out the solution in some cases, for instance, when the initial or the final densities take values that approach zero. Indeed, an important limitation of the method is its inability to handle densities that reach zero somewhere in the domain. A solution to this problem would be to perturbate the density by adding a small quantity everywhere on the space then rescale the result. However, even for densities that are no where null, but takes values that are close to zero, the method becomes numerically instable. The convergence becomes very slow, and the quality of the solution degrades sensibly. As numerical experiments demonstrate, adding the term $H\,|q|^2$ helps accelerating the convergence process and smoothing the solution. Since the new problem is a perturbated version of the original, the obtained solution has a sub-optimal objective value, and the deviation from optimality increases with larger values of H. This implies that H must be selected to strike a compromise between optimality and convergence speed. Section 4.3 shows experimentally the effects of the smoothing term on the behavior of the algorithm.

2.4. *Parallelization*

The only global step in the the algorithm proposed in [Benamou and Brenier, 2000] is the solution to the Poisson PDE. Upon discretization, this equation is transformed to a sparse linear system that can be solved using a number of parallel methods [Saad, 2001]. In the constrained case, computing μ creates an additional interdependence between the data, the degree of which depends on the nature of the constraint. In the simplest case, the constraint is pointwise and no interdependence is added. The constraint can as well be spacial, in which case computing μ can be divided to a set of independent problems, one for each time instant. The case where the constraint is space-time is the hardest to parallelize, since it causes the greatest degree of interdependence between the problem data. Parallelism in the optimal transport problem is therefore data driven, which implies that the computational domain has to be distributed among the processors. Optrans offers three possible distribution schemes:

- *Partitioned*: in this scheme, the domain is divided as equally as possible among the processors. This is intended to be used with space domains before the time dimension is added. Time is added so that all the points in the time-space domain corresponding to the same point in the space domain are located in the same processor (Figure 1).
- *Time-sliced*: in this scheme, a space-time domain is split along the time dimension, with each processor being responsible for part of the time interval. Every processor keeps the whole space domains corresponding to the time instants belonging to its part (Figure 2). This scheme is mainly intended for constrained transport problems with a spacial constraint. Indeed, the scarcity of distributed quadratic optimization

solvers makes partitioning the space domain an inefficient strategy, since the data corresponding to the whole space must be moved to the same location to solve the quadratic problem.

– *Duplicated*: in this scheme, the domain is duplicated on all processors. This is intended for space domains that are used to generate other domains. For instance, a space domain is duplicated before being transformed into a time-space domain and sliced along the time dimension.

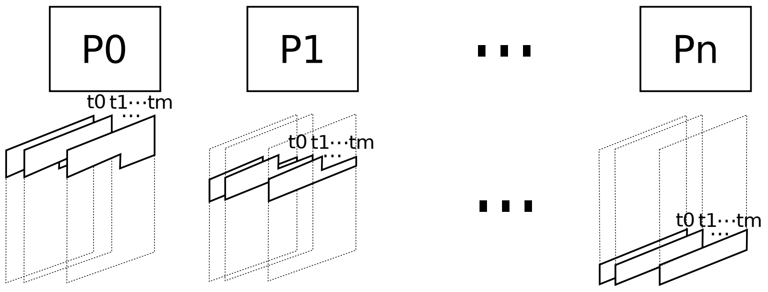

Fig. 1. Distribution of a time-space domain over a partitioned space domain among n processors. m is the time resolution.

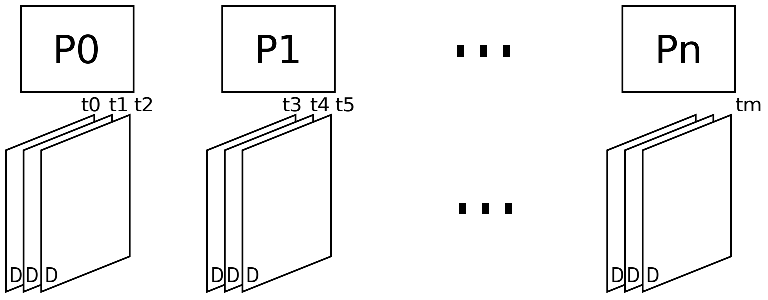

Fig. 2. Distribution of a time-sliced domain among n processors. D is the space domain, and m is the time resolution.

3. The Optrans Library

The current version of Optrans implements solvers for free and convexly constrained problems on regular grid box domains of arbitrary dimensions discretized by finite difference. Each dimension of the domain can have cell or vertex-centered discretization, and its boundary conditions can be periodic or null Neumann. The software is developed in C++ and currently tested under Linux. From the design perspective, three main objectives are considered:

(i) Exploiting parallelism: solving optimal transport problems requires high computational power and large amounts of memory. Parallelism is therefore necessary for solving large scale problems. On the other hand, the rapid development of multi-core processors and their availability makes parallelization a rational choice to exploit the computational power that is already available to the users even if they are only concerned with small or medium size problems.

(ii) Providing an easy interface: optimal transport is a field that regroups researchers from different backgrounds and with various degrees of familiarity with programming and the use of optimization libraries. Designing a simple interface is therefore necessary to ensure the accessibility of the library to the largest possible segment of the research community.

(iii) Ensuring extendability: scientific computing literature is rich with different approaches to representing computational domains and discretizing partial differential equations, which obviously can not be implemented conjointly at the initial stage. Consequently, the library must be designed with extendability in mind in order to incorporate new representations and discretization schemes with minimum effort and alteration to the existing code.

Optrans makes use of the object oriented capabilities of C++, such as as inheritance and polymorphism, to achieve these objectives. In what follows, the internal architecture and the programming interface of Optrans are presented.

3.1. *Internal architecture*

The design of Optrans follows an object-oriented approach, with objects representing entities that are naturally related to the problem. For instance, the object *Domain* represents the computational domain, *Function* represents a function, and *VField* represents a vector field. Figure 3 shows an overview of the interaction between these objects. The interfaces through which this interaction takes place are kept as generic as possible in order to hide as much as possible the internal functioning of each object. This is achieved by using the mechanisms of inheritance and virtual methods overloading.

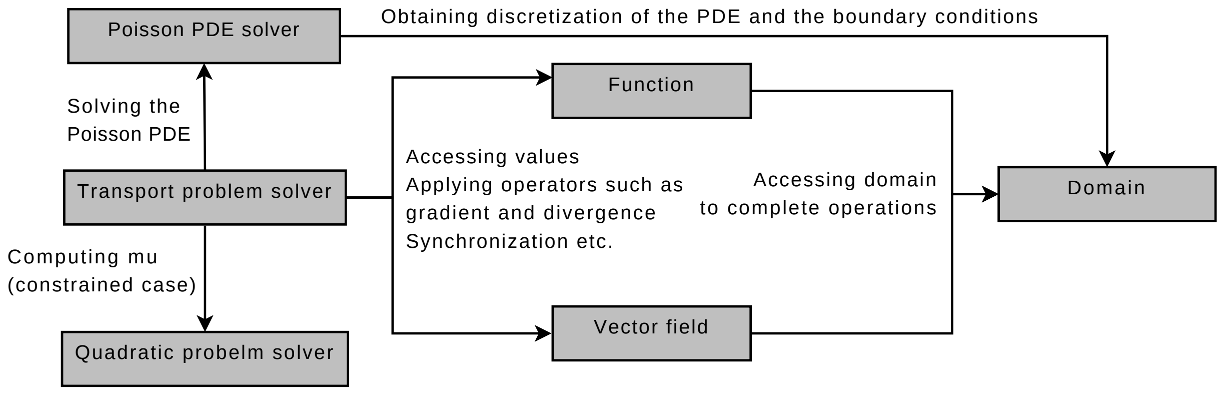
Poisson PDE solver
Obtaining discretization of the PDE and the boundary conditions
Function
Solving the Poisson PDE
Accessing values
Applying operators such as gradient and divergence
Synchronization etc.
Accessing domain to complete operations
Transport problem solver
Domain
Computing mu (constrained case)
Vector field
Quadratic probelm solver

Fig. 3. Overview of the interaction between the internal objects of the library.

At the functional level, Optrans makes use of a number of external libraries to complete its task (Figure 4). Optrans uses MPI (Message Passing Interface) for message passing and synchronization, which allows it to run on distributed as well as shared memory architectures. MPI is a mature specification with implementations available on virtually all parallel architectures, from networks of personnel computers to super computers. Its widespread use makes it the de facto standard for parallel applications. Using MPI allows Optrans to benefit from existing parallel libraries, for instance the hypre library as explained below, and also facilitates its potential integration as part of other parallel applications.

The BoomerAMG solver [Henson and Yang, 2002], which is part of the hypre library [Falgout et al., 2006a], is used for solving the linear system resulting from the Poisson PDE. hypre is library of high performance preconditioners and solvers dedicated to solving large, sparse linear systems on massively parallel computers. Besides its mature development status, which is the result of years of refinement, hypre has the advantage of offering different types of interfaces to suit the different needs of applications [Falgout et al., 2006b]. Of these interfaces, Optrans uses what is called the Linear-Algebraic Interface or the IJ interface for short. This is basically the most generic interface, where the matrix and the right hand side vector are passed to hypre, then a solver is chosen to solve the corresponding linear system. The matrix and the right hand side vector are given to the library in distributed form, that is, every processor stores part of the matrix and vector. Optrans discretizes the Poisson equation, handles the boundary conditions and prepares the coefficients of the matrix and the right hand side vector before passing them through the IJ interface. The choice of the IJ interface is due to its flexibility and the fact that it is not limited in term of the dimensionality of the problem. The BoomerAMG solver is a parallel implementation of algebraic multigrid, which offers various choices for coarsening and relaxation techniques, and because of its general purpose nature, it represents a natural choice for using with Optrans.

In the constrained case, Optrans solves the arising quadratic problem using either the Ipopt solver [Wächter and Biegler, 2006], with the user providing the constraints, or a function entirely provided by the user. Ipopt is a library for large-scale nonlinear optimization. It computes the local solutions of optimization problems having the from

$$\begin{cases} \min\limits_{x\in\mathbb{R}^n} f(x)\,; \\ \text{s.t. } g_L \le g(x) \le g_U \text{ and } x_L \le x \le x_U, \end{cases}$$

where the constraints are vector valued. From the performance perspective, using the Mehrotra algorithm [Mehrotra, 1992] implemented in Ipopt is faster than using the interior point method of Ipopt. At the implementation level, Ipopt is written in C++, and therefore, it can be easily interfaced with Optrans. Although not a distributed optimizer, Ipopt can be configured to use a parallel linear solver, such as Pardiso [Schenk et al., 2001] or MUMPS [Amestoy et al., 2000], that can take advantage of shared memory architectures.

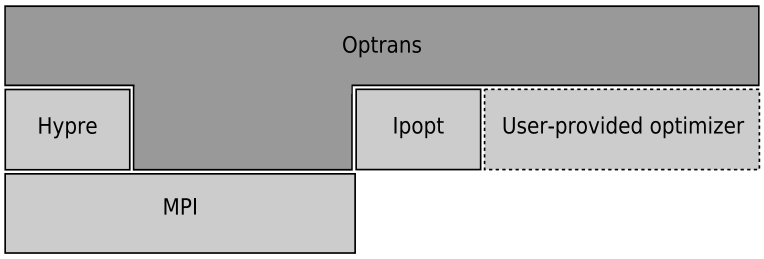

Fig. 4. Optrans and external libraries.

3.2. *Programming interface*

The library is accessed via a set of classes representing domains, functions, vector fields, transport problems and solvers. The approach followed is to separate between the description of the problem and the algorithms used to solve it, which is similar in spirit to [Meza et al., 2007], for instance. The user first creates a representation of the problem, then instantiates a solver object to which the problem representation is passed. The user can choose the algorithm and the parameters that the solver uses to solve the problem. To illustrate the use of Optrans programming interface, a simple example is presented. Consider an optimal transport problem in a regular two dimensional box with periodic boundary conditions. In the following, the steps necessary to solve such a problem with Optrans are presented.

First, and like any MPI based program, initialization is needed.

```
/* Initialize MPI */
MPI_Init(&argc, &argv);
```

After initializing MPI, the computational domain must be defined. For this, the dimension, the mesh resolution, the mesh step size, the type of boundary conditions and the type of discretization are to be specified. The domain is created then distributed among the processors. The distribution scheme used in this example is *Partitioned*, which can be set by calling the method *partition* on the domain object.

```
int dim= 2;
int meshRes= 32;
double meshStep= 1.0/32;
BCType bCTypes[dim];
DiscType discTypes[dim];
bCTypes[1]= bCTypes[0]= Periodic;
discTypes[1]= discTypes[0]= CellCentered;
/* Create the domain */
RGCube dom(MPI_COMM_WORLD, dim, bCTypes, discTypes, meshRes, meshStep);
/* Partition the domain */
dom.partition();
```

The next step is to create the transport problem object. This requires specifying the domain, the initial and the final densities. The latter are objects of type *Function* defined on the computational domain. In the distribution scheme of this example, only the local part of the function values needs to be passed. The local interval is given by the properties *ilower* and *iupper* of the domain object.

```
rhoInit= new Function(&dom);
rhoFinal= new Function(&dom);
ilower= dom.getIlower();
iupper= dom.getIupper();
for(k=ilower;k<=iupper;k++)
{
        (*rhoInit)[k]= ...;
        (*rhoFinal)[k]= ...;
}
/* Create the transport problem */
OptransPb optransPb(&dom,rhoInit,rhoFinal);
```

Once the transport problem object is created, it is passed to a solver object along with the time resolution. The solver is controlled by a number of parameters, which are divided into three major groups. The first group contains the parameters that control the behavior of the algorithms. These are set using the *setAlgParams* method. In the present example, the algorithm that is run by the solver is *ALG0*, which denotes the algorithm proposed in [Benamou and Brenier, 2000]. For this algorithm, the only relevant parameters are r and the smoothing parameter *H*. The second group of parameters specifies the stopping criteria. These are the maximum number of iterations, the maximum allowed residual, the maximum allowed normalized criterion (see [Benamou and Brenier, 2000]), the maximum allowed change in mass and the maximum allowed change in μ. The algorithm is stopped whenever any one of these criteria is verified. These parameters are set using *setTolParams*. The third group of parameters, which is not shown in this example, controls the I/O operations during the execution, for instance, enabling or disabling trace, the trace file name, the frequency of backup points and the backup file name. This group of parameters is set using the *setIOParams* method.

```
/* Create the solver */
OptransCFDSolverC solver(&optransPb,timeRes);
/* Set algorithm */
solver.setAlgParams(ALG0, r, s, H, rho, rhoR, rhoS);
/* Set tolerance */
solver.setTolParams(maxIter, maxRes, maxNCr, maxDeltaMass, maxDeltaMu);
```

At this stage, all the parameters are set, and the only remaining step is to initialize the solver and then call the *solve* method.

```
/* Initialize */
solver.init();
/* Solve */
solver.solve();
```

Finally, once computed, the solution to the problem can be obtained from Optrans through arguments, or saved to disk. Information about the execution, such the number of iterations or the residual, can also be obtained by calling the appropriate methods on the solver object.

The classes *Function* and *VField* offer the possibility of reading and writing their values from and to file, which can be useful for inputting and outputting data to and from Optrans. There are two available modes for this operation, centralized and distributed:

– In the centralized mode, the processor with ID 0 reads the whole set of values from file and distributes the content to each other processor. This can be useful for simplifying the storage of the data or in the case where some processors have no input/output capabilities.
– In the distributed mode, each processor reads its portion of the data. Therefore, each of the portions has to be stored in a separate file. This mode is useful for computer networks, where each processor has a local storage device.

4. Experimental Results

This section presents a set of numerical experiments to demonstrate the working of Optrans and the numerical methods used therein. Experiment 1 shows a three dimensional example problem that can be solved by Optrans. In Experiment 2, a transport problem with null Neumann boundary conditions is solved. In Experiment 3, the effect of the smoothing term $H\,|q|^2$ are studied. Finally, Experiment 4 shows the effect of parallelization on the execution time.

4.1. *Experiment 1: Solving a three dimensional problem*

The first experiment demonstrates the use of Optrans by solving a free transport problem on a three dimensional domain with periodic boundary conditions. The domain is discretized by a cell-centered mesh with a resolution of $20 \times 20 \times 20$. Figure 5 shows the density and the velocity field of the computed solution at different time instants. Figure 6 shows the evolution of the objective function and the convergence criteria. The results confirms that, indeed, the optimality conditions for the problem are satisfied, and that the obtained solution offers a smooth interpolation between the initial and the final densities.

4.2. *Experiment 2: Null Neumann boundary conditions*

Figure 7 shows the solution of a transport problem on a domain with periodic boundary conditions. The concentration of mass initially at the left is divided into two parts, one part is transported to the right, and the other is transported through the left boundary. Figure 8 shows the solution to the same transportation problem, but this time on a domain with null Neumann boundary conditions. Here, the mass can not be transported through the boundary, hence the whole concentration of mass is transported within the domain towards the right side.

4.3. *Experiment 3: The effects of the smoothing term*

The goal of this experiment is to show the effects of the smoothing term on the quality of the solution and the convergence process. To this end, the initial and the final densities are chosen to have values that approach zero. For the sake of this experiment, both densities reach a minimum value of approximately 0.005. Figure 9 shows the solution computed without any smoothing, that is H=0. The solution has steep, non smooth transitions, and there are clear distortions at the initial and the final densities. On the other hand, Figure 10 shows the solution with H=0.5. Clearly, the solution is much smoother and there are no visible distortions at the initial and the final time instants.

The effect of the parameter H on the convergence of the algorithm is presented in Figure 11, 12, 13 and 14, which show the evolution of the objective function and the convergence criteria for the values of H =0, 0.1, 0.5 and 1 respectively. It can be observed that as H gets smaller, the gap between the kinetic energy and the product $\langle \mu, q \rangle$ becomes smaller, the fluctuations in the values of the energy also diminish, and lower values of kinetic energy are attainable. On the other hand, the minimum residual attainable becomes larger, and similarly is the case for $|\nabla\phi - q|$. Furthermore, small values of H cause slow convergence as $|\nabla\phi - q|$ decreases more slowly, causing in turn the slow convergence of μ. Large values of H offer quick convergence, but the solution has a higher objective value. The graphs show that, in this example, an intermediate value of H that offers a good compromise between smoothness of the solution and rapid convergence on the one hand and the quality of the solution on the other hand is H=0.5.

4.4. *Experiment 4: Parallelization*

The goal of this experiment is to show the speedup obtained by parallelizing the execution of the algorithms. Three test problems are used:

(i) A free problem: this is an ordinary transport problem without any constraints.

(ii) A transport problem with bound: in this problem, each point in the space has an upper bound on its density. The upper bound may vary from one point to another, and the goal is transport the initial density to the final one without violating any bound constraint.
(iii) A metric deformation problem: the initial and the final densities in this case are distance functions. The transport must be done so that the intermediate densities are all distance functions as well (see [Kerrache and Nakauchi, 2010]).

Note that the problems are cited in order of increasing complexity, with the free problem being the simplest and the metric deformation problem being the most complex.

Figure 15 shows the execution time for the three test problems as the number of processors varies. For this experiment, the solver is run on a multi-core machine (shared memory). The graphs show that there is reduction in execution time in all cases, and that the speedup increases with the problem complexity. The reason is that, because of the constraints being spacial, the quadratic problem necessary for computing μ can be split into smaller problems, one for each time instant, that can be solved simultaneously. This reduces the execution time considerably, especially when the constraint is complex.

5. Conclusion

This paper introduced Optrans, a parallel library for solving the time-dependent optimal transport problem. Both free and convexly-constrained problems can be handled by the library. Optrans is designed using an object oriented approach and developed in C++. It offers a simple, object-oriented programming interface. The current version supports regular grid box domains discretized by finite difference. A number of techniques introduced in the library were presented, including solving transport problems on domains with null Neumann boundary conditions and handling densities that approach zero using a smoothing term. The parallelization of the processing shows gain in the execution time, especially for spacially constrained problems with complex constraints. As a next step in the development of Optrans, the possibility of handling more complicated geometries and discretization schemes, namely, structured grids and finite element domains in two and three dimensions, is considered. The hypre interfaces for structured grids and finite elements can provide an important help in implementing these improvements.

References

[Amestoy et al., 2000] Amestoy, P. R., Duff, I. S., and L'Excellent, J. Y. (2000). Multifrontal parallel distributed symmetric and unsymmetric solvers. *Computer Methods in Applied Mechanics and Engineering*, 184(2-4):501 – 520.

[Benamou and Brenier, 2000] Benamou, J.-D. and Brenier, Y. (2000). A computational fluid mechanics solution to the Monge-Kantorovich mass transfer problem. *Numer. Math.*, 84(3):375–393.

[Brenier et al., 2003] Brenier, Y., Frisch, U., Hénon, M., Loeper, G., Matarrese, S., Mohayaee, R., and Sobolevskiĭ, A. (2003). Reconstruction of the early Universe as a convex optimization problem. *Mon. Not. R. Astron. Soc.*, 346:501–524.

[Carrillo et al., 2007] Carrillo, J., Di Francesco, M., and Toscani, G. (2007). Strict contractivity of the 2-Wasserstein distance for the porous medium equation by mass-centering. *Proc. Am. Math. Soc.*, 135(2):353–363.

[Cohen and Kundu, 2004] Cohen, I. M. and Kundu, P. K. (2004). *Fluid Mechanics, Third Edition.* Academic Press.

[Cullen and Maroofi, 2003] Cullen, M. and Maroofi, H. (2003). The Fully Compressible Semi-Geostrophic System from Meteorology. *Archive for Rational Mechanics and Analysis*, 167:309–336.

[Cullen, 2006] Cullen, M. J. P. (2006). *A Mathematical Theory of Large-scale Atmosphere/ocean Flow.* Imperial College Press.

[Falgout et al., 2006a] Falgout, R., Jones, J., and Yang, U. (2006a). The design and implementation of hypre, a library of parallel high performance preconditioners. In Barth, T. J., Griebel, M., Keyes, D. E., Nieminen, R. M., Roose, D., Schlick, T., Bruaset, A. M., and Tveito, A., editors, *Numerical Solution of Partial Differential Equations on Parallel Computers*, volume 51 of *Lecture Notes in Computational Science and Engineering*, pages 267–294. Springer Berlin Heidelberg.

[Falgout et al., 2006b] Falgout, R. D., Jones, J. E., and Yang, U. M. (2006b). Conceptual interfaces in hypre. *Future Gener. Comput. Syst.*, 22(1-2):239–251.

[Fortin and Glowinski, 1983] Fortin, M. e. and Glowinski, R. e. (1983). *Augmented Lagrangian methods: Applications to the numerical solution of boundary-value problems.* Studies in Mathematics and its Applications, 15. Amsterdam-New York-Oxford: North-Holland. XIX, 340 p.

[Frisch et al., 2002] Frisch, U., Matarrese, S., Mohayaee, R., and Sobolevski, A. (2002). A reconstruction of the initial conditions of the Universe by optimal mass transportation. *Nature*, 417:260–262.

[Gangbo and McCann, 2000] Gangbo, W. and McCann, R. J. (2000). Shape recognition via wasserstein distance. *Q. Appl. Math.*, LVIII(4):705–737.

[Glowinski and Tallec, 1989] Glowinski, R. and Tallec, P. L. (1989). *Augmented Lagrangian and operator-splitting methods in nonlinear mechanics.* SIAM.

[Haker et al., 2004] Haker, S., Zhu, L., Tannenbaum, A., and Angenent, S. (2004). Optimal mass transport for registration and warping. *Int. J. Comput. Vision*, 60(3):225–240.

[Henson and Yang, 2002] Henson, V. E. and Yang, U. M. (2002). Boomeramg: a parallel algebraic multigrid solver and preconditioner. *Appl. Numer. Math.*, 41(1):155–177.

[Jenkins and Traub, 1970] Jenkins, M. and Traub, J. (1970). A three-stage variable-shift iteration for polynomial zeros and its relation to generalized Rayleigh iteration. *Numer. Math.*, 14:252–263.

[Jordan et al., 1998] Jordan, R., Kinderlehrer, D., and Otto, F. (1998). The variational formulation of the fokker-planck equation. *SIAM J. Math. Anal.*, 29(1):1–17.

[Kantorovitch, 1942] Kantorovitch, L. (1942). On the translocation of masses. *C. R. (Dokl.) Acad. Sci. URSS, n. Ser.*, 37:199–201.

[Kerrache and Nakauchi, 2010] Kerrache, S. and Nakauchi, Y. (2010). Constrained optimal transport and metric deformation. Submitted.

[Mehrotra, 1992] Mehrotra, S. (1992). On the implementation of a primal-dual interior point method. *SIAM Journal on Optimization*, 2(4):575–601.

[Meza et al., 2007] Meza, J. C., Oliva, R. A., Hough, P. D., and Williams, P. J. (2007). Opt++: An object-oriented toolkit for nonlinear optimization. *ACM Trans. Math. Softw.*, 33(2):12.

[Museyko et al., 2009] Museyko, O., Stiglmayr, M., Klamroth, K., and Leugering, G. (2009). On the application of the monge-kantorovich problem to image registration. *SIAM J. Img. Sci.*, 2(4):1068–1097.

[Rachev and Ruschendorf, 1998] Rachev, S. T. and Ruschendorf, L. (1998). *Mass Transportation Problems: Volume II: Applications (Probability and Its Applications).* Springer.

[Saad, 2001] Saad, Y. (2001). Parallel iterative methods for sparse linear systems. In Dan Butnariu, Y. C. and Reich, S., editors, *Inherently Parallel Algorithms in Feasibility and Optimization and their Applications*, volume 8 of *Studies in Computational Mathematics*, pages 423 – 440. Elsevier.

[Schenk et al., 2001] Schenk, O., Grtner, K., Fichtner, W., and Stricker, A. (2001). Pardiso: a high-performance serial and parallel sparse linear solver in semiconductor device simulation. *Future Generation Computer Systems*, 18(1):69 – 78.

[Villani, 2009] Villani, C. (2009). *Optimal transport. Old and new.* Grundlehren der Mathematischen Wissenschaften 338. Berlin: Springer. .

[Wächter and Biegler, 2006] Wächter, A. and Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57.

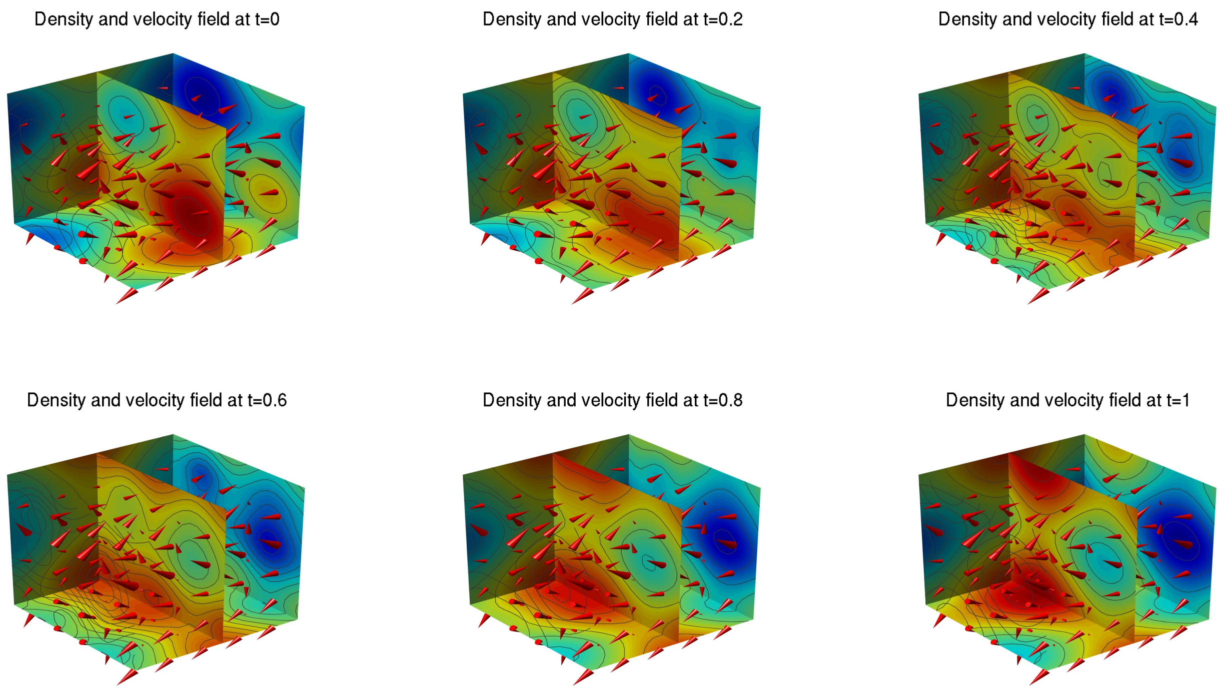
Density and velocity field at t=0
Density and velocity field at t=0.2
Density and velocity field at t=0.4
Density and velocity field at t=0.6
Density and velocity field at t=0.8
Density and velocity field at t=1

Fig. 5. A three dimensional transport problem.

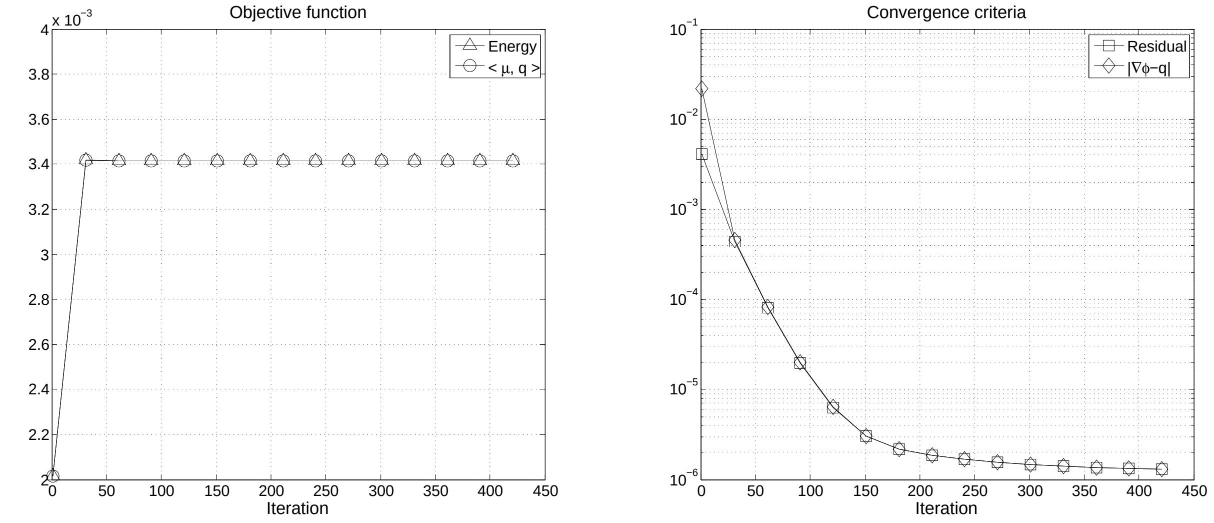

Fig. 6. Evolution of the objective function and the convergence criteria for the three dimensional problem.

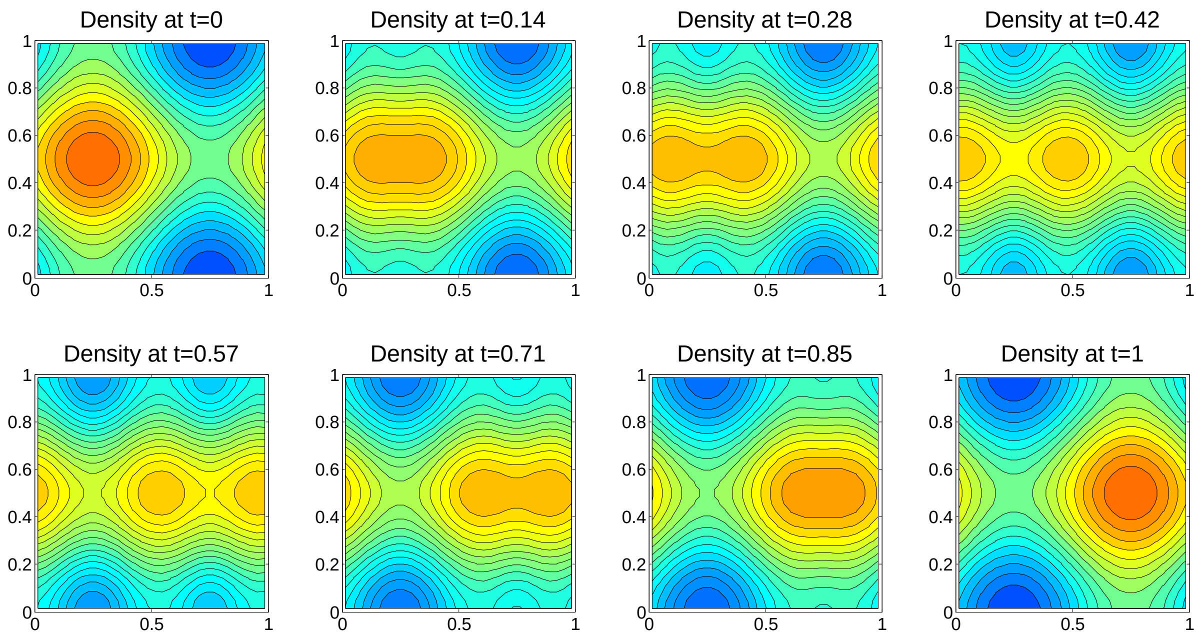

Fig. 7. Transport on a domain with periodic boundary conditions.

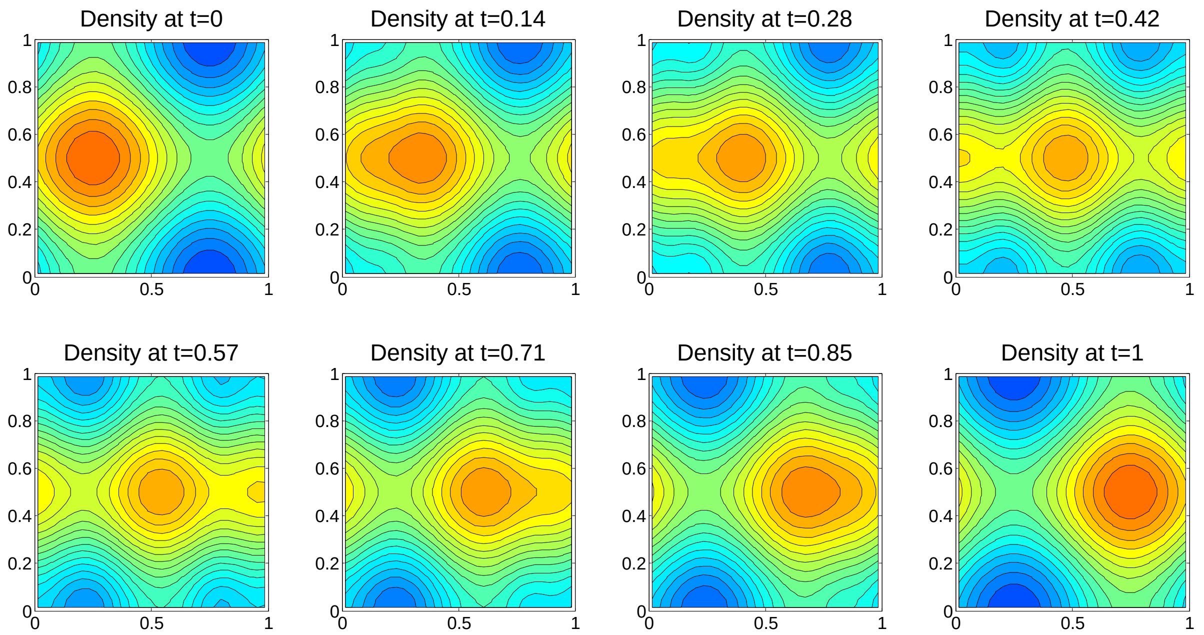

Fig. 8. Transport on a domain with null Neumann boundary conditions.

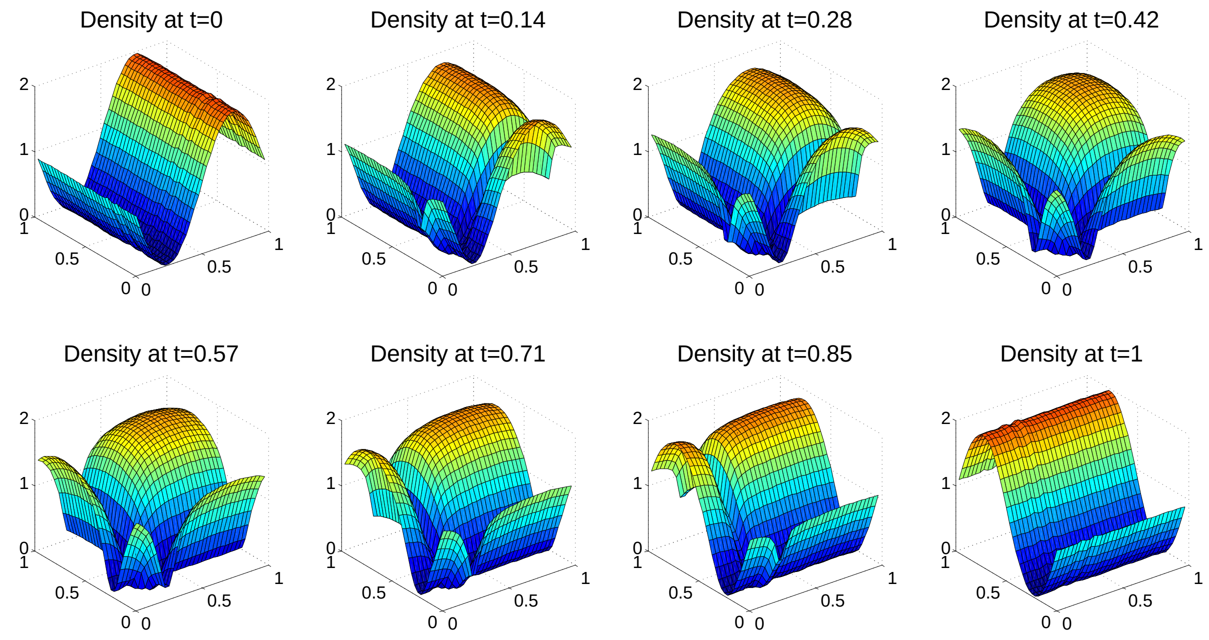

Fig. 9. Transport of a density that reaches zero without smoothing (H=0).

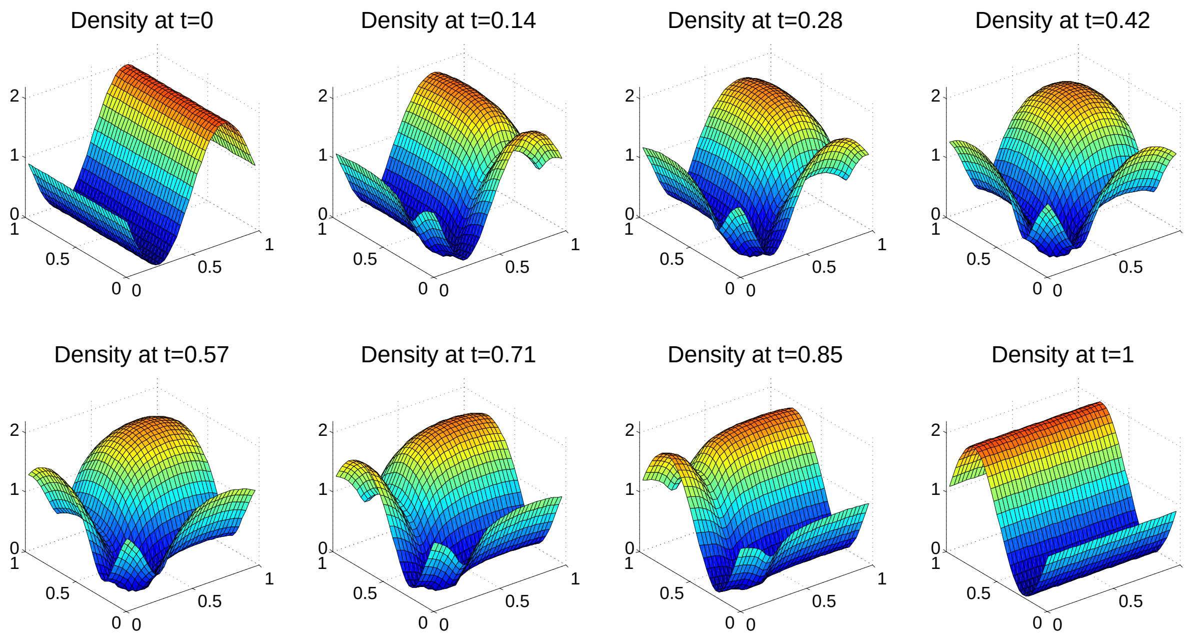

Fig. 10. Transport of a density that reaches zero with H= 0.5.

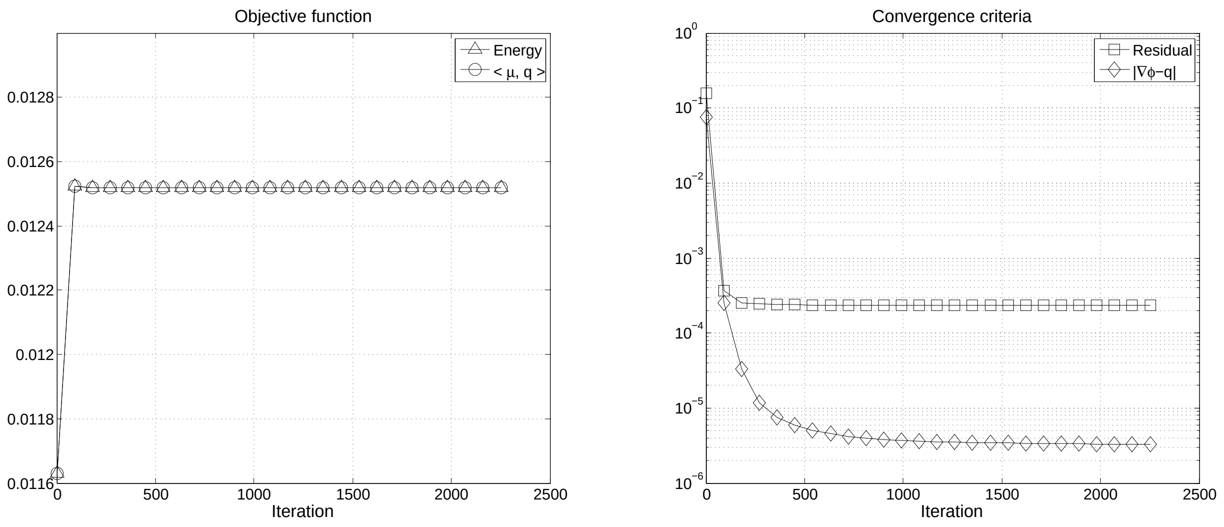

Fig. 11. Evolution of the objective function and the convergence criteria for H=0.

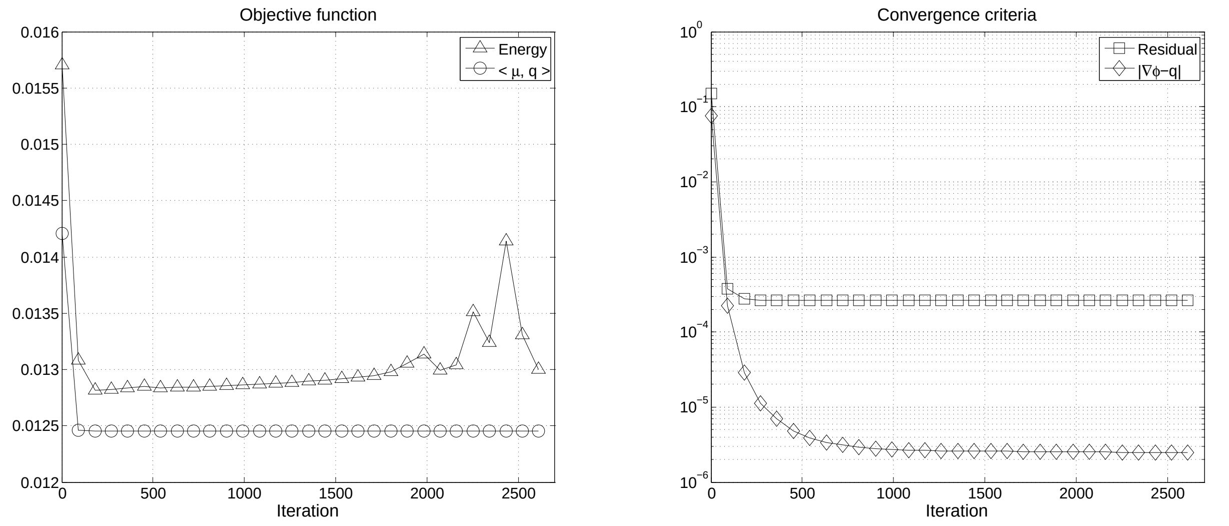

Fig. 12. Evolution of the objective function and the convergence criteria for H=0.1.

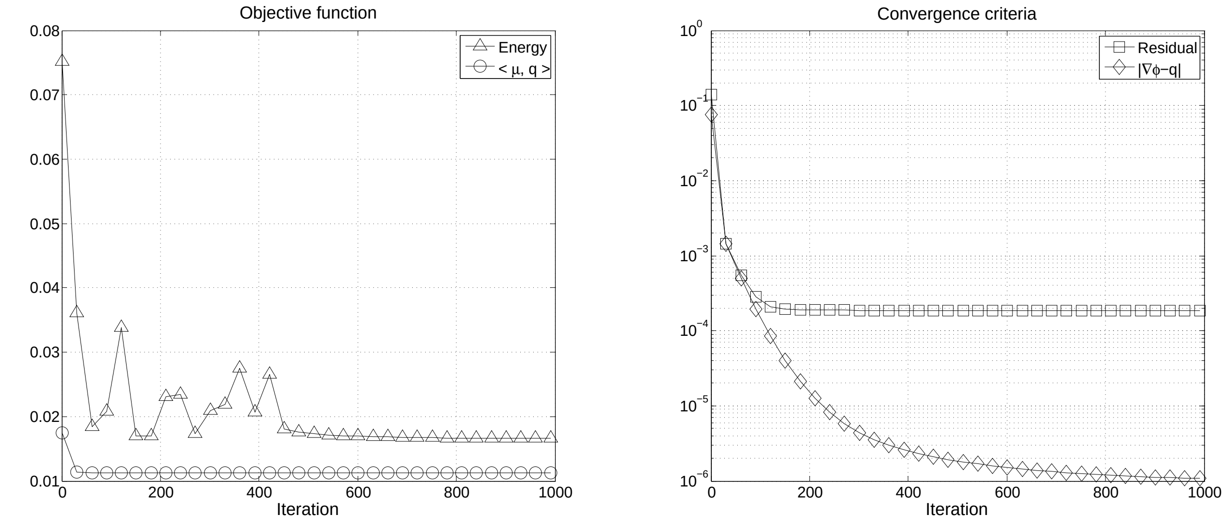

Fig. 13. Evolution of the objective function and the convergence criteria for H=0.5.

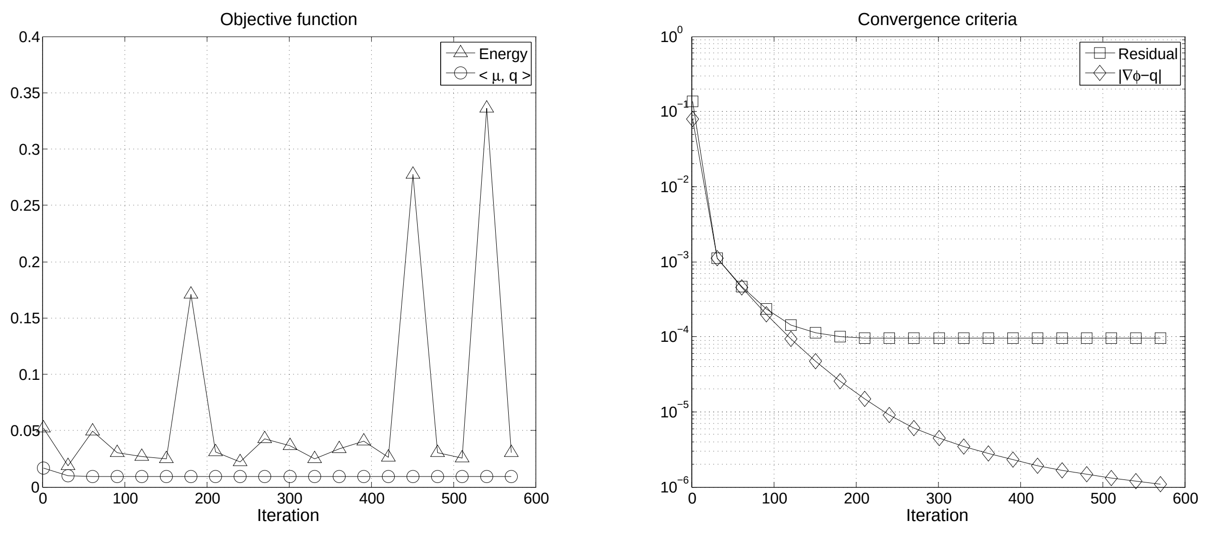

Fig. 14. Evolution of the objective function and the convergence criteria for H=1.

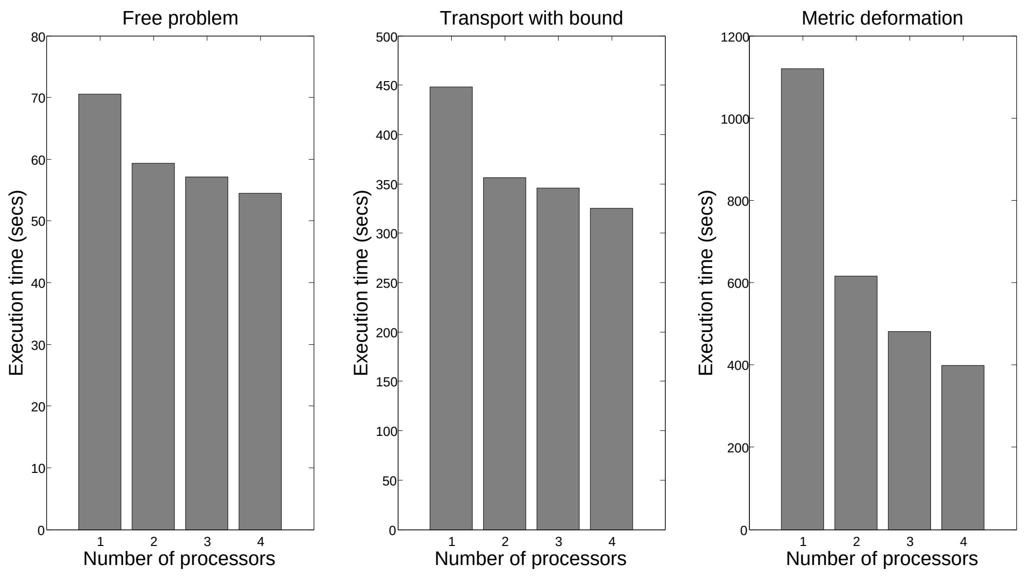

Fig. 15. Execution time on a multi-core architecture.