J Sci Comput
DOI 10.1007/s10915-014-9894-1

Fast Methods for Computing Centroidal Voronoi Tessellations

James C. Hateley · Huayi Wei · Long Chen

Received: 14 February 2012 / Revised: 17 January 2014 / Accepted: 15 July 2014

Abstract A Centroidal Voronoi tessellation (CVT) is a Voronoi tessellation in which the generators are the centroids for each Voronoi region. CVTs have many applications to computer graphics, image processing, data compression, mesh generation, and optimal quantization. Lloyd's method, the most widely method used to generate CVTs, converges very slowly for larger scale problems. Recently quasi-Newton methods using the Hessian of the associated energy as a preconditioner are developed to speed up the rate of convergence. In this work a graph Laplacian preconditioner and a two-grid method are used to speed up quasi-Newton schemes. The proposed graph Laplacian is always symmetric, positive definite and easy to assemble, while the Hessian, in general, may not be positive definite nor easy to assemble. The two-grid method, in which an optimization method using a relaxed stopping criteria is applied on a coarse grid, and then the coarse grid is refined to generate a better initial guess in the fine grid, will further speed up the convergence and lower the energy. Numerical tests

The first author was supported by 2010–2011 UC Irvine Academic Senate Council on Research, Computing and Libraries (CORCL) award. The second author was supported by NSFC (Grant No. 11301449), in part by Specialized Research Fund for the Doctoral Program of Higher Education (Grant No. 20134301120003). The third author was supported by National Science Foundation (NSF) (DMS-0811272 and DMS-1115961), in part by 2009–2011 UC Irvine Academic Senate Council on Research, Computing and Libraries (CORCL) award, and in part by Department of Energy prime award # DE-SC0006903.

J. C. Hateley
Irvine, CA, USA
e-mail: hateleyjc@gmail.com

H. Wei
Hunan Key Laboratory for Computation and Simulation in Science and Engineering,
School of Mathematics and Computational Science, Xiangtan Uinversity, Xiangtan 411105,
Hunan, People's Republic of China
e-mail: huayiwei1984@gmail.com

L. Chen (✉)
Department of Mathematics, University of California, Irvine, Irvine, CA 92697, USA
e-mail: chenlong@math.uci.edu

Published online: 19 August 2014

show that our preconditioned two-grid optimization methods converges fast and has nearly linear complexity.

Keywords Centroidal Voronoi tessellation · Lloyd's method · Numerical optimization · Quasi-Newton methods

Mathematics Subject Classification 62H30 · 65H10 · 65K10 · 65U05 · 68U10

1 Introduction

Let $\Omega \subset \mathbb{R}^n$ be an open domain and $\mathbf{z} = \{z_i\}_{i=1}^N \subset \Omega$ be a specified set of points. A *Voronoi Tessellation* $\mathcal{V} = \{V_i\}_{i=1}^N$ is a special type of partition of Ω such that each region V_i, the so-called *Voronoi region*, is defined by

$$V_i = \{x \in \Omega : |x - z_i| < |x - z_j| \text{ for } j \neq i\}. \tag{1}$$

The points $\mathbf{z}$ are called *generators*. A *Centroidal Voronoi Tessellation* (CVT) is a Voronoi tessellation where each generator z_i coincides with the centroid of each Voronoi region V_i. CVTs have many applications to computer graphics, image processing, data compression, mesh generation, and optimal quantization; see [9,12] for specific details.

We consider fast methods for computing stable CVTs. A CVT can be also defined through the energy associated to a set of generators $\mathbf{z}$ and the corresponding Voronoi tessellation $\mathcal{V}$

$$\mathcal{E}(\mathbf{z}, \mathcal{V}(\mathbf{z})) = \sum_{i=1}^{N} \mathcal{E}_i(\mathbf{z}, \mathcal{V}(\mathbf{z})) = \sum_{i=1}^{N} \int_{V_i} \rho(x) \|x - z_i\|^2 \, dx, \tag{2}$$

where ρ is a probability density function. It can be easily shown that a CVT is a critical point of $\mathcal{E}$ and a stable CVT is defined as a local minimal point of $\mathcal{E}$. In other words, we are interested in fast methods on finding a local minimizer of the energy $\mathcal{E}(\mathbf{z}, \mathcal{V}(\mathbf{z}))$.

Du and Emelianenko [6] developed a quasi-Newton method mixed with Lloyd iterations for solving the gradient equation $\nabla \mathcal{E} = 0$. This approach works well on finding a CVT provided a sufficient number of Lloyd iterations are applied [27]. Du and Emelianenko [7] also describe a Newton-based multilevel algorithm and later on Emelianenko [13] developed a multigrid method by using ideas from algebraic multigrid methods. These methods are effective as a root finding method for solving the gradient equation $\nabla \mathcal{E} = 0$, but may suffer on not leading to a stable CVT; see numerical examples in [27].

One promising method to find a stable CVT is developed by Liu, Wang, Lévy et al. [27], where preconditioned quasi-Newton methods are applied to minimize the energy $\mathcal{E}$. The preconditioner used in [27] is an incomplete Chloesky factorization of a modification of the Hessian matrix of the energy $\mathcal{E}$. However, there is no guarantee the modified Hessian will be an effective preconditioner, especially when large and/or numerous shifts are needed; see [1,25].

The main contribution of this paper is to propose a graph Laplacian as a preconditioner. The graph Laplacian we propose is easy to assemble and always symmetric and positive definite (SPD). Therefore the incomplete Cholesky factorization algorithm can be applied directly. Numerical tests show that classical quasi-Newton methods and nonlinear conjugate gradient methods coupled with our preconditioner converges with a rate almost independent of the problem size and thus has nearly linear complexity. Note that finding a good preconditioner for a specific problem is a central question in the scientific computing [36].

Another contribution is the combination of the two-grid method [35] with our preconditioned optimization methods. Running our preconditioned optimization methods on the coarse grid and prolongate to a fine grid, the energy is much lower than random generators or running Lloyd's method for an equivalent time on the fine grid. This makes for a good initial guess on the fine grid, which allows for faster convergence and often leads to a lower energy. Since the run-time on the coarse grid is significantly less than that in the fine grid, it further speeds up our preconditioned optimization methods.

The paper is organized as follows. In Sect. 2, we review the problem, previous works and well known optimization methods. In Sect. 3, we derive our graph Laplacian. In Sect. 4, we show one-dimensional examples using a quasi-Newton scheme. In Sect. 5, we provide several numerical examples with both preconditioned and non-preconditioned optimization methods. In Sect. 6, we show numerical results applying a two-grid with both preconditioned and non-preconditioned optimization methods. Lastly in Sect. 7, we summarize the results with concluding remarks.

2 Preliminaries

In this section we present the background of the problem and briefly review well known optimization methods. Recall that our object of interest is the energy functional (2). Taking derivative to $\mathbf{z}$ and noting that the part $\frac{\partial \mathcal{E}}{\partial \mathcal{V}}(\mathbf{z}, \mathcal{V}) = 0$ due to the definition of the Voronoi region, we can get:

$$\mathcal{F}_i(\mathbf{z}) := \frac{\partial}{\partial z_i}\mathcal{E}(\mathbf{z}, \mathcal{V}(\mathbf{z})) = 2\int_{V_i} \rho(x)(z_i - x)\, dx. \tag{3}$$

Solving $\mathcal{F}_i(\mathbf{z}) = 0$, we get

$$z_i = \left(\int_{V_i} \rho(x)\, dx\right)^{-1} \int_{V_i} x\rho(x)\, dx. \tag{4}$$

Namely z_i is the centroid of the Voronoi region V_i with respect to the density ρ. Therefore a CVT is a critical point of $\mathcal{E}(\mathbf{z})$. A *stable CVT* will be defined as a local minimizer of the energy $\mathcal{E}(\mathbf{z}, \mathcal{V})$, while an *unstable CVT* corresponds to a saddle point.

Before moving to various optimization methods, we mention that there are plenty of resources for forming Voronoi tessellations for a given set of generators. *Computational Geometry Algorithms Library (CGAL)* [15] is one such. Another is MATLAB with the command `voronoin`. We choose to implement our code on the latter. To construct a Voronoi tessellation on a bounded convex domain, we reflect all the nodes, which are close enough to the boundary, over the boundary. From this expanded set of nodes we form the dual graph, known as the Delaunay triangulation. Once this triangulation is made we connect the circumcenters of neighboring triangles, then truncate back to the original domain which results the Voronoi tessellation of a bounded domain. The complexity of generating the Voronoi tessellation of N generators is $\mathcal{O}(N \log N)$ in 2-D and 3-D; see [38].

For every generator z_i and its Voronoi region V_i, we triangulate V_i by connecting z_i and the vertices of V_i, see Fig. 1b. Then we can use Gaussian quadrature on triangles to compute integrals in $\mathcal{E}$ and $\nabla\mathcal{E}$. We also note that the energy is in the order of $N^{-2/d}$ [19,20] for optimal CVTs in $\mathbb{R}^d$. A fair stopping tolerance for different N would be size-independent. Therefore we introduce $\mathcal{D} := \text{diag}(m_i)$, where $m_i = \int_{V_i} \rho(x)\, dx$, the diagonal matrix formed by the

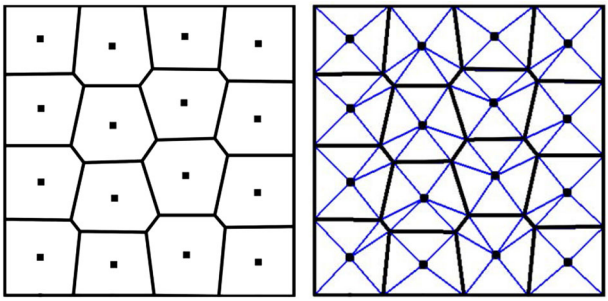

Fig. 1 Decomposition of a Voronoi region into triangles. **a** Voronoi regions on a square with 12 generators. **b** A triangulation of the Voronoi region

mass of every Voronoi region, and the scaled l_2 norm $\|\mathcal{D}^{-1}\mathcal{F}\|$. Here for the sake of notation we define $\mathcal{F}(\mathbf{z}) := \nabla\mathcal{E}(\mathbf{z}, \mathcal{V}(\mathbf{z}))$.

2.1 Lloyd's Method

The most popular method for computing a CVT is Lloyd's method [29], which treat the regions and generators as independent and update them alternatively. Namely fix $\mathbf{z}$ and optimize the partition which is the voronoi tessellation $\mathcal{V}(\mathbf{z})$, and then fix $\mathcal{V}$ and move $\mathbf{z}$ to the centers.

Let us introduce the so-called Lloyd map $\mathbf{T} = (T_1, T_2, \ldots, T_N)$ [7]. Given $\mathbf{z} = \{z_i\}_{i=1}^N$ and corresponding Voronoi regions $\mathcal{V} = \{V_i\}_{i=1}^N$, T_i is defined by

$$T_i(z_i) = \left(\int_{V_i} \rho(x)\,dx\right)^{-1} \int_{V_i} \rho(x)x\,dx, \tag{5}$$

for $i = 1, 2, \ldots, N$. For a CVT, the generator $\mathbf{z}$ is a fixed point of the Lloyd map and the Lloyd iteration is a fixed point iteration using the Lloyd map.

Given an initial set of generators $\mathbf{z}_0$, one Lloyd iteration is as follows:

Lloyd Iteration

1. Construct $\mathcal{V}_k(\mathbf{z}_k)$.
2. Update $\mathbf{z}_{k+1} = \mathbf{T}(\mathbf{z}_k)$.

Lloyd's method is easy to implement and has a convergence rate of $1 - \mathcal{O}(h^2)$, where $h = \min_i \operatorname{diam}(V_i)$ [8,14]. Thus the larger the size of the problem is, the slower the rate of convergence is. In each iteration of Lloyd's method the construction of $\mathcal{V}$ dominates the computation time.

2.2 Quasi-Newton Methods

Recently Liu et al. [27] has shown that the energy $\mathcal{E}$ has second order smoothness for convex domains, because of this quasi-Newton methods seem reasonable. It should be noted that second order smoothness, regardless of the domain, is only possible away from degeneracies [37].

Given B an approximation of the Hessian of $\mathcal{E}$, a quasi-Newton iteration is as follows:

quasi-Newton Iteration

1. Solve $B\delta\mathbf{z} = -\mathcal{F}(\mathbf{z}_k)$.
2. Update $\mathbf{z}_{k+1} = \mathbf{z}_k + \alpha_k \delta\mathbf{z}$.

Here α_k is given by the line search which satisfies the strong Wolfe conditions [31]. If $B = \mathcal{H}(\mathbf{z}_k)$, the Hessian of the energy at $\mathbf{z}_k$, and $\alpha_k = 1$, then it returns to the classic Newton's method. When the Hessian is indefinite, which creates a problem in Newton's method, one remedy is to modify $\mathcal{H}$ by diagonal shifting, then an incomplete Cholesky factorization is applied for the modified Hessian; see Algorithm 3.1 in [25].

2.3 Quasi-Newton Method: PLBFGS

In quasi-Newton methods, the Hessian is approximated by successive gradient vectors. One of the most popular quasi-Newton schemes is BFGS method (Broyden–Fletcher–Goldfarb–Shanno) [31]. We only describe one iteration of the preconditioned limited memory version of the BFGS (PLBFGS), and refer to [26,31] for more details.

After initializing $r = -\mathcal{F}(\mathbf{z}_0)$ and two integer parameters M and T, each iteration of the PLBFGS algorithm is as follows:

PLBFGS(M,T) iteration

1. First LBFGS update:
   for $i = \min(M-1, k-1) : -1 : 1$
      Calculate $\gamma_i = \rho_i s_i^t r$
      Update residual $r = r - \gamma_i y_i$
   end
2. Set search direction:
   if $k \bmod T = 0$ solve $A_k d_k = r$
   else $d_k = H_k r$
   end
3. Second LBFGS update:
   for $i = 1 : +1 : \min(M-1, k-1)$
      Update search direction $d_k = d_k + s_i(\gamma_i - \rho_i y_i^t d_k)$
   end
4. Update $\mathbf{z}_{k+1} = \mathbf{z}_k + \alpha_k d_k$

The matrix A_k is a chosen preconditioner, α_k is given by the line search satisfying the weak Wolfe conditions [31] with an initial guess $\alpha_k = 1$, and

$$s_k = \mathbf{z}_{k+1} - \mathbf{z}_k,\ y_k = \mathcal{F}_{k+1} - \mathcal{F}_k,\ \rho_k = \left(y_k' s_k\right)^{-1},\ H_k = \frac{s_{k-1}' y_{k-1}}{y_{k-1}' y_{k-1}}.$$

2.4 Preconditioned Nonlinear Conjugate Gradient (PNLCG)

The nonlinear conjugate gradient method (NLCG) is a generalization of the conjugate gradient method for solving linear algebraic equation to nonlinear optimization problems. We outline the basic iteration step and refer to [21,31] for details.

Let A_0 be a symmetric positive definite (SPD) matrix. Given an initial guess $\mathbf{z}_0$ and $p_0 = -A_0^{-1}\mathcal{F}(\mathbf{z}_0)$, one iteration of the PNLCG scheme from $\mathbf{z}_{k-1}$ to $\mathbf{z}_k$, for $k = 1, 2, \ldots$, is as follows:

PNLCG Iteration

1. Update $\mathbf{z}_k = \mathbf{z}_{k-1} + \alpha_{k-1} p_{k-1}$.
2. Solve $A_k \mathbf{x}_k = -\mathcal{F}(\mathbf{z}_k)$.
3. Calculate β_k.
4. Update conjugate direction $p_k = \mathbf{x}_k + \beta_k p_{k-1}$.

There are several formulae for updating β_k; see [21]. Three well-known β_k are: Polak-Ribière (β_k^{PR}) [32], Hestenes–Stiefel (β_k^{HS}) [22], Fletcher–Reeves (β_k^{FR}) [17]. In our experiments, β_k^{PR} shows to be a good choice for this problem. The matrix A_k is often chosen as an SPD and known as a preconditioner. A good preconditioner will speed up convergence dramatically.

There are many ways to solve the equation $A_k \mathbf{x}_k = -\mathcal{F}(\mathbf{z}_k)$. We report results using an incomplete Cholesky factorization (ICF); see Algorithm 3.1 in [25]. We note that algebraic multigrid (AMG) methods [2,28,33] with the optimal complexity $\mathcal{O}(N)$ will be ideal for large scale problems in 3-D. We have tested AMG in [3], but due to the implementation using MATLAB it is slightly slower than the built-in ICF in MATLAB.

The scalar α_k is obtained by line search such that the strong Wolfe conditions is satisfied [31]. In each line search step, the evaluation of energy and its gradient request the computation of Voronoi diagram which is relatively expensive. A good initial guess can dramatically decrease the number of searching steps, the number of iterations of NLCG, and thus save the overall computational time. For CVT optimization, our tests show that, in most cases, choosing $\alpha_k^0 := \mathcal{F}(\mathbf{z}_k)' A_k^{-1} \mathcal{F}(\mathbf{z}_k) / |p_k' \mathcal{H}(\mathbf{z}_k) p_k|$, the strong Wolfe condition will be satisfied. Where $\mathcal{H}(\mathbf{z}_k)$ is the Hessian at $\mathbf{z}_k$.

It is good to note that LBFGS iterations with $M = 1$ coincides NLCG iterations with β^{HR}. NLCG is thus more memory efficient than LBFGS (with $M > 1$).

3 A Graph Laplacian

In this section we derive a graph Laplacian that carries some information about the Hessian, but is inexpensive to construct and invert. This work is motivated by the recent work with Optimal Delaunay Triangulations; see [4,5].

Recall that if a smooth function $u : \mathbb{R}^n \to \mathbb{R}$ satisfies the mean value property $u(x) = \frac{1}{|B(x,r)|} \int_{B(x,r)} u(y)\, dy$ for all ball $B(x, r)$ centered at x with radius $r > 0$, then u is harmonic, i.e., $\Delta u = 0$. The discretization of Δu will lead to Au with a discrete Laplacian matrix. One can change the Lebesgure measure to a more general one $\rho(x)dx$ and the metric in defining the ball in the mean value property. The corresponding A will be a different weighted discrete Laplacian matrix.

For a CVT, the generator $\mathbf{z}$ is a fixed point of the Lloyd map (5), i.e. $z_i = \frac{1}{|V_i|_\rho} \int_{V_i} \rho(y) y\, dy$ which can be interpreted as a discrete mean value property for function $u(x) = x$. Thus we would expect $A(\mathbf{z})\mathbf{z} = 0$ with a discrete Laplacian matrix A. In other words, we are interested in finding such a matrix equation to approximate the non-linear equation $\mathcal{F}(\mathbf{z}) = 0$.

Before we get into details, we remark that the matrix of an effective preconditioner is not necessarily a good approximation to the Hessian matrix. A SPD preconditioner can be thought as a change of coordinate such that the descent method (gradient or conjugate–gradient method) in the new coordinate is effective. The search direction will be corrected by the quasi-Newton or conjugate gradient method. When searching for an effective preconditioner, a rough approximation, e.g., first order approximation of $\mathcal{F}(\mathbf{z})$ is often times acceptable. We now derive such an approximation.

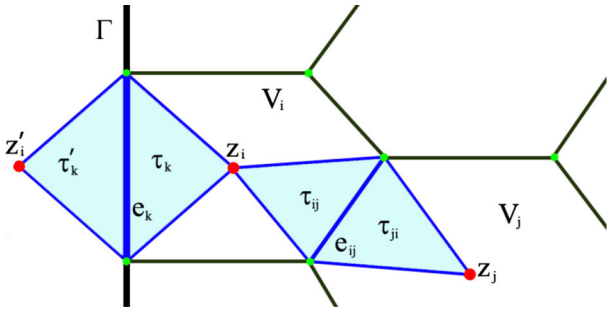

Fig. 2 Decomposition of a Voronoi region for the construction of a graph Laplacian

For a polygon domain $\Omega \in \mathbb{R}^2$ and a set of generators, it is known that every Voronoi region is a polygon. Given a generator z_i and its Voronoi region V_i, if e_{ij} is a common edge shared by V_i and another Voronoi region V_j, connecting z_i and z_j to the two end points of e_{ij}, respectively, we can get two triangles τ_{ij} and τ_{ji}; see Fig. 2. If e_k is edge of V_i and $e_k \subset \Gamma := \partial\Omega$, we get only one triangle τ_k.

Let $\mathcal{J}_i$ be the index set of the generators whose Voronoi regions are adjacent to V_i and τ_{ij} be the triangle formed from the generator z_i and the edge $\overline{V}_i \cap \overline{V}_j$; see Fig. 2. Then $\mathcal{F}(\mathbf{z}) = 0$ can be written as

$$z_i \int_{V_i} \rho(x)\, dx - \sum_{j\in\mathcal{J}_i} \int_{\tau_{ij}} \rho(x) x\, dx = 0 \tag{6}$$

We approximate the left hand side of Eq. (6) by a linear average of its respective neighbors using a first order approximation of the integrals. We first approximate ρ by a constant ρ_c and integrals by one point quadrature

$$\int_{\tau_{ij}} \rho(x) x\, dx \approx \rho_c \frac{z_i + z_j}{2} |\tau_{ij}|.$$

Here the point $(z_i + z_j)/2 \in e_{ij} \subset \tau_{ij}$ since e_{ij} is the bisector of edge $z_i z_j$ by the definition of Voronoi region. Note that this approximation is still valid in three and higher dimension. If we further approximate the weighted area $\rho_c|\tau_{ij}|$ by the integral $b_{ij} = \int_{\tau_{ij}} \rho(x)\, dx$, which is still first order accurate, then we obtain an approximated first order equation

$$\sum_{j\in\mathcal{J}_i} b_{ij}(z_i - z_j) = 0. \tag{7}$$

However the matrix B formed by b_{ij} is non-symmetric, i.e., $b_{ij} \neq b_{ji}$; see Fig. 2. A symmetry version will be

$$a_{ii} z_i + \sum_{j\in\mathcal{J}_i} a_{ij} z_j = 0,$$

where the weight is constructed as follows:

$$A = \begin{cases} a_{ij} = -\int_{\tau_{ij}\cup\tau_{ji}} \rho(x)\, dx & i \neq j \\ a_{ii} = \sum_{e_{ij}=\overline{V}_i\cap\overline{V}_j} |a_{ij}| + 2\sum_{e_k=\Gamma\cap\overline{V}_i} \int_{\tau_k} \rho(x)\, dx & \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

For the off diagonals a_{ij} is the opposite of the mass of τ_{ij} and τ_{ji}. The diagonal entries a_{ii} are the sum of absolute values of the off diagonal entries in the ith row. For the generators contain the edges $e_k \subset \Gamma$, for each triangle τ_k, the factor of 2 comes about by a reflection to preserve the boundary of the domain, see $\mathbf{z}_i^{'}$ and $\tau_{ij}^{'}$ in Fig. 2. Note that the factor 2 can be multiplied to (6) so that $\int_{\tau_{ij}\cup\tau_{ji}}$ can be thought as an approximation of $2\int_{\tau_{ij}}$. In short $A = B + B^t$. Such change enables the matrix A is symmetric.

It is easy to observe that, by construction, A is symmetric and diagonal dominant and thus a SPD. The incomplete Cholesky factorization algorithm can be applied directly without diagonal shifting [25]. Other fast solvers, for example, algebraic multigrid methods [2,28,33] with the optimal complexity $\mathcal{O}(N)$ can be also used to invert the graph Laplacian. More specifically, the Lean Algebraic Multigrid (LAMG) Fast Graph Laplacian Linear Solver [28] supports symmetric diagonally dominant matrices with non-zero-row sums. Our graph Laplacian is an ideal example of such a matrix. We have tested our 2-D problems using both AMG and ICF but only report results using the ICF. Due to the implementation we use in MATLAB AMG performs slightly slower than the built-in function `ichol`. We expect in 3-D, using a similar construction of our graph Laplacian on tetrahedrons, AMG will be faster than an ICF.

Our graph Laplacian is easier to construct. The quantity $\int_{\tau_{ij}} \rho(x)\,dx$ has been computed during the evaluation of $\mathcal{E}$ and $\nabla\mathcal{E}$. In contrast the Hessian matrix of the energy functional $\mathcal{E}$ is complicated and relatively expensive to compute. For completeness we include the formulate of the Hessian below

$$D^2\mathcal{E} = \begin{cases} \frac{\partial^2 E}{\partial z_i^{(k)}\partial z_i^{(k)}} = 2m_i - \sum_{j\in\mathcal{J}_i} \frac{2}{\|z_i - z_j\|}\int_{e_{ij}} (z_i^{(k)} - x^{(k)})^2\rho(x)d\sigma \\ \frac{\partial^2 E}{\partial z_i^{(k)}\partial z_i^{(l)}} = -\sum_{j\in\mathcal{J}_i} \frac{2}{\|z_i - z_j\|}\int_{e_{ij}} (z_i^{(k)} - x^{(k)})(z_i^{(l)} - x^{(l)})\rho(x)d\sigma & k \neq l \\ \frac{\partial^2 E}{\partial z_i^{(k)}\partial z_j^{(l)}} = \frac{2}{\|z_i - z_j\|}\int_{e_{ij}} (z_i^{(k)} - x^{(k)})(z_j^{(l)} - x^{(l)})\rho(x)d\sigma & j \in \mathcal{J}_i \\ \frac{\partial^2 E}{\partial z_i^{(k)}\partial z_j^{(l)}} = 0 & \text{otherwise} \end{cases}$$

Note that our graph Laplacian will be applied to each coordinate component of the generator which can be interpreted as using the same coordinate transformation for each coordinate component.

4 Numerical Results: One Dimension

We start by looking at results from quasi-Newton iterations for one dimensional CVTs. We test two non-trivial density functions and compare the results to the Full Approximation Scheme (FAS) method described in Koren, Yaveneh and Spira [24].

4.1 Description of Methods

Given an interval $\Omega = (a, b)$ and generators $\mathbf{z}$, the Voronoi regions are simplified to $V_i = (d_{i-1}, d_i) = \left(\frac{z_{i-1}+z_i}{2}, \frac{z_i+z_{i+1}}{2}\right)$. The Hessian is a tridiagonal matrix whose components are given by

$$\frac{\partial^2 \mathcal{E}}{\partial z_i \partial z_{i-1}} = -\frac{1}{2}\rho(d_{i-1})\,(z_i - z_{i-1})\,, \quad \frac{\partial^2 \mathcal{E}}{\partial z_i \partial z_{i+1}} = -\frac{1}{2}\rho(d_i)\,(z_{i+1} - z_i)\,,$$
$$\frac{\partial^2 \mathcal{E}}{\partial z_i^2} = 2\int_{d_{i-1}}^{d_i} \rho(x)\,dx - \frac{1}{2}\rho(d_i)(z_{i+1} - z_i) - \frac{1}{2}\rho(d_{i-1})(z_i - z_{i-1}). \tag{9}$$

Because of the simplicity of the problem in one dimension, we only show quasi-Newton iterations using our graph Laplacian. In one dimension our graph Laplacian is a modified Hessian by replacing the diagonal of Hessian matrix by the sum of the absolute values of off-diagonals, with suitable modification near the boundary generators; that is,

$$A = \text{diag}\left(\frac{\partial^2 \mathcal{E}}{\partial z_i \partial z_{i-1}}, \left|\frac{\partial^2 \mathcal{E}}{\partial z_i \partial z_{i-1}}\right| + \left|\frac{\partial^2 \mathcal{E}}{\partial z_i \partial z_{i+1}}\right|, \frac{\partial^2 \mathcal{E}}{\partial z_i \partial z_{i+1}}\right). \tag{10}$$

We also implement a two-grid method. Starting on a coarse grid, we run an optimization method using a relaxed stopping criteria and then refine the coarse grid by consecutive midpoints to the fine grid.

Two-grid and quasi-Newton Method for 1-D CVT

1. Initialize $\mathbf{z}_0$ on the coarse grid.
2. Run Lloyd's method until relaxed stopping criteria is met.
3. Refine to fine grid by successive uniform refinements.
4. Preform quasi-Newton iterations using the graph Laplacian until stopping criteria is met.

For the coming numerical examples, we fix the coarse grid at 32 uniformly distributed generators. We preform Lloyd's method until $\|\mathcal{D}^{-1}\mathcal{F}(\mathbf{z})\| < 1.\text{e}{-4}$. We refine the coarse grid by consecutive midpoints upto the desired fine grid. We use a simple backtracking line search to ensure that generators do not move outside the domain. For both tests we use a fifth order Gaussian quadrature for numerical integration.

The number of generators in the finest grid is 2^L, where L ranges from 7 to 16. The stopping criteria is set to $\|\mathcal{D}^{-1}\mathcal{F}(\mathbf{z})\| < 1.\text{e}{-6}$. The level L, iteration steps *Iter*, energy $\mathcal{E}$, and l_2-norm $\|\mathcal{D}^{-1}\mathcal{F}(\mathbf{z}_k)\|$ are summarized in tables. The decay of the weighted gradient and the l_2-norm of the residual $\|\delta z\|$, the solution of $A\delta z = -\mathcal{F}(\mathbf{z}_k)$, are shown in figures. For these tests we also calculate the average residual reduction factor (RRF), which is given by $\|\delta z_{k+1}\|\backslash\|\delta z_k\|$, and compare with [24].

Table 1 Data table for example 1, density $\rho(x) = e^{-10x^2}$ on $(-1, 1)$

L	Iter	$\mathcal{E}$	$\|\mathcal{D}^{-1}\mathcal{F}\|$	L	Iter	$\mathcal{E}$	$\|\mathcal{D}^{-1}\mathcal{F}\|$
7	10	1.3264e−04	1.3307e−07	12	14	1.3348e−07	7.7409e−09
8	10	3.3677e−05	2.2935e−07	13	15	3.3387e−08	3.8520e−09
9	11	8.4849e−06	7.3432e−08	14	15	8.3489e−09	6.6885e−07
10	12	2.1295e−06	3.2315e−08	15	16	2.0875e−09	3.3436e−07
11	13	5.3341e−07	1.5651e−08	16	17	5.2190e−10	1.6716e−07

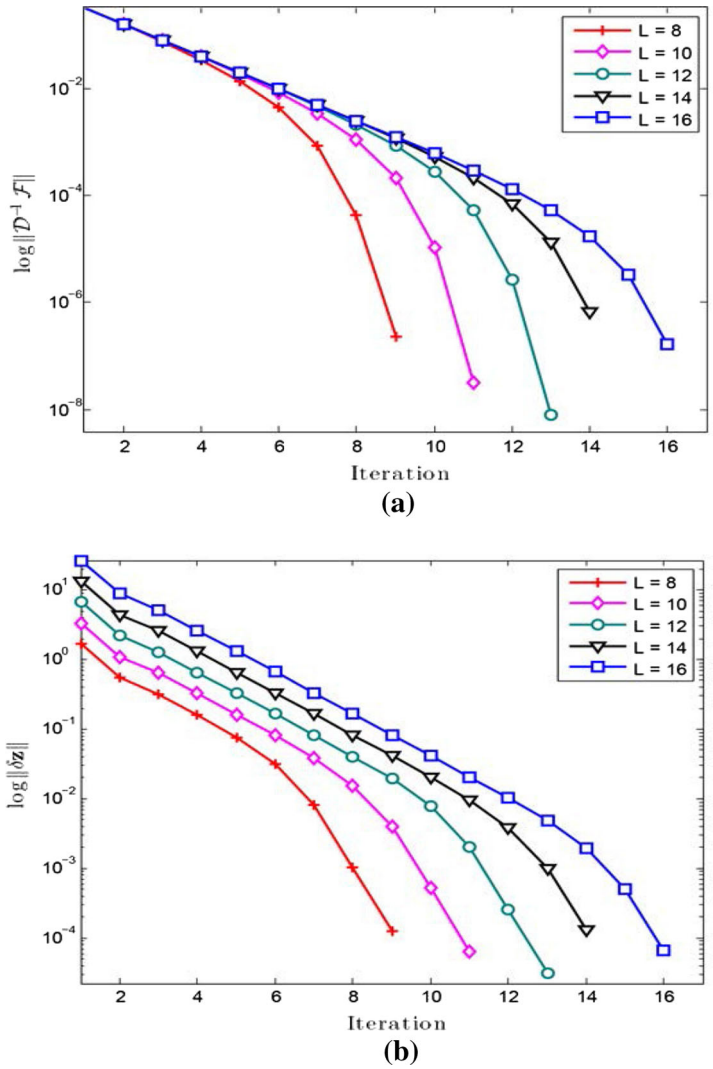

Fig. 3 Error and residual figures for example 1, density $\rho(x) = e^{-10x^2}$ on $(-1, 1)$. **a** Iteration versus log of the l_2-norm of weighted error. **b** Iterations versus log of l_2-norm of residual

4.2 Example 1

First we test the Gaussian distribution $\rho(x) = e^{-10x^2}$ on the domain $\Omega = (-1, 1)$. As the level increases, there is a slight increase in the number of iterations. This can be seen from Table 1. For the first several iterations, the quasi-Newton method using our graph Laplacian shows linear convergence. Afterwards superlinear convergence is obtained, which is an evidence that our graph Laplacian becomes a better and better approximation of the Hessian; see Fig. 3a. The average RRF is 0.3927.

The coarse grid is fixed at 32 generators. As the fine grid gets finer and finer, the initial guess obtained by interpolation from the fixed coarse grid is less accurate.

Table 2 Data table for example 2, density $\rho(z) = 6x^2e^{-2x^3}$ on $(0, 2)$

L	Iter	$\mathcal{E}$	$\|\mathcal{D}^{-1}\mathcal{F}\|$	L	Iter	$\mathcal{E}$	$\|\mathcal{D}^{-1}\mathcal{F}\|$
7	12	1.0288e−05	4.9381e−08	12	16	1.0124e−08	4.5549e−08
8	12	2.5822e−06	7.5461e−07	13	17	2.5312e−09	2.2802e−08
9	13	6.4680e−07	3.7076e−07	14	17	6.3284e−10	7.1679e−07
10	14	1.6186e−07	1.8329e−07	15	18	1.5821e−10	3.5830e−07
11	15	4.0484e−08	9.1485e−08	16	19	3.9554e−11	1.7918e−07

Therefore on average it takes one more iteration to enter the superlinear convergence region and consequently one more iterations to reach the stopping criteria. One could improve the performance (e.g. the iteration steps are uniform to the levels) by using three-grids or multi-grids version. But for the one dimensional problem, the triangular graph Laplacian matrix can be efficiently inverted and the construction of Voronoi tessellation is trivial, the saving is not significant. For simplicity, we only present the two-grid version.

4.3 Example 2

For the second example we test the Weibull distribution $\rho(x) = 6x^2e^{-2x^3}$ on $\Omega = (0, 2)$. There is a slight oscillation during the first few iterations in the residual, this raises the average RRF to 0.5248. The oscillation in the residual decreases when superlinear convergnce is obtained. Again on average it takes approximately one more iteration per level to see this behavior on the residual (Table 2).

The weighted error shows linear convergence and then superlinear convergence; see Fig. 4a. The results are consistent with example 1. Typically as the levels increase it takes, on average, one more iteration for our method to obtain superlinear convergence.

The two numerical examples show that the two-grid method is effective and the overall effectiveness of the quasi-Newton scheme with the proposed graph Laplacian. The order at which the error converges superlinearly, for both examples for the convergence, is approximately 1.8. This is not quite the quadratic convergence of Newton's method; however, this is expected since we are not using the Hessian but our graph Laplacian. We note that in [24], the RRF of FAS is 0.17–0.20, while the RRF of our method is between 0.39 and 0.52. However, the FAS presented in [24] for one dimension problem, the authors believe, as well as we do, is difficult to be extend to higher dimensions [23]. In contrast, our methodology works well in two dimensions and possible higher dimensions.

5 Numerical Results: Two Dimensions

In this section we present numerical results for generating two dimensional CVT using NLCG and BFGS with our graph Laplacian preconditioner. We compare the results of the preconditioned optimization methods versus their non-preconditioned counter parts and versus Lloyd's method.

We use the l_2-norm of the scaled gradient $\|\mathcal{D}^{-1}\mathcal{F}\|$ to measure the convergence and set our stopping criteria at $\|\mathcal{D}^{-1}\mathcal{F}\| <$1.e−6. In practice, if the energy is the main concern,

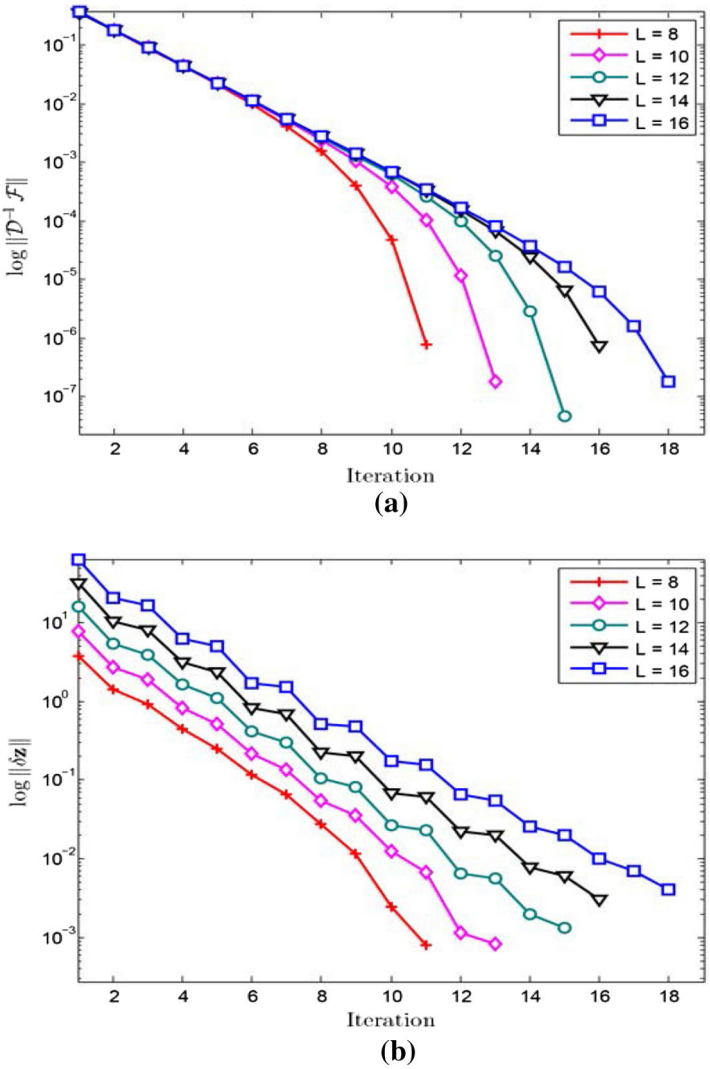

Fig. 4 Figure for error and residual for example 2, density $\rho(z) = 6x^2 e^{-2x^3}$ on $(0, 2)$. **a** Iteration versus log l_2-norm of the weighted error. **b** Iterations versus log of l_2-norm of residual

one can set the iteration criteria to an insufficient decrease in the energy, i.e. $|\mathcal{E}_k - \mathcal{E}_{k-1}| <$ tolerance. We shall comment on the steps for energy to level off.

For various methods, we compare the iteration steps and CPU time to reach the stopping criteria. For Lloyd's method, we only count the first 1,000 iterations since it takes too many to reach the same stopping criteria. Since the generation of Voronoi diagram takes significant time, we use *nFeval* to count the number of evaluations of the energy function in which a new Voronoi diagram is generated.

For each test we start with 2,000 random distributed generators based on the given probability density function. Results will be different for different initial guesses. We thus test the sensitivity of each method with respect to the initial guess by running twenty simulations and

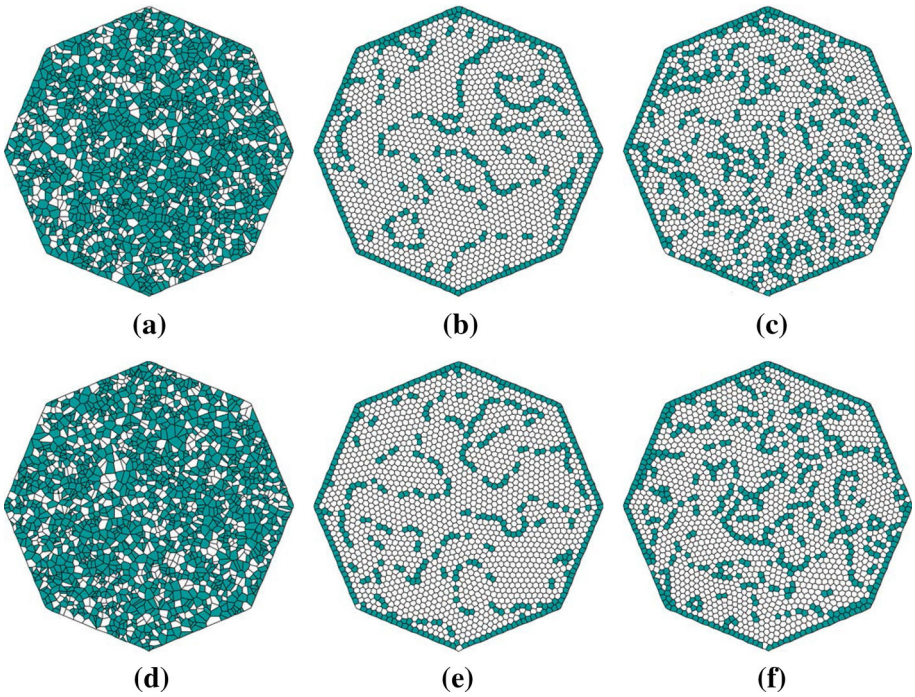

Fig. 5 Voronoi tessellations for example 3. **a** An initial Voronoi tessellation with random generators, **b** CVT obtained from a PLBFGS(7) using **a**, **c** CVT obtained from 1,000 iterations of Lloyd's method using **a**, **d** an initial Voronoi tessellation with random generators, **e** CVT obtained from a PLBFGS(7) using **d**, **f** CVT obtained from 1,000 iterations of Lloyd's method using **d**

recording the performance of each simulation, see Figs. 6, 9 and 13. For each optimization method from these 20 simulations we compare the averages of results.

To test the complexity, we change N, the number of generators, from 2,000 to 8,000 with an increment of 1,000 and plot the computational time versus N. For each fixed number of generators and optimization method, we run each test twenty times and use the averaged results.

For a CVT, most Voronoi regions seem to form hexagons [16,18]. For the figures of Voronoi tessellations we color the Voronoi regions to illustrate this, see Figs. 5, 8 and 12. The white Voronoi regions are hexagons while the shaded Voronoi regions are other polygons.

Each Voronoi region V_i is triangulated by connecting its generator z_i and the vertices of V_i. Gaussian quadrature on triangles is applied to compute $\mathcal{E}$ and $\nabla\mathcal{E}$.

In our tests, we use the optimization packages HANSO and NLCG developed by Overton (http://www.cs.nyu.edu/overton/software/) and the finite element method package *i*FEM [3] by Chen (http://www.math.uci.edu/~chenlong/programming.html). Note that we write our code in *MATLAB*, while Liu et al. [27] use *C++*. Thus a direct comparison of computational time of our methods versus theirs is not feasible. Instead we use the Lloyd's method with fixed 1,000 iterations as the base to compare.

We run all simulations on a Laptop running Ubuntu 11.10. The CPU is an Intel i5-2410M Processor with 3M Cache at 2.30 GHz and 4 GB of DDR SDRAM at 1,333 MHz.

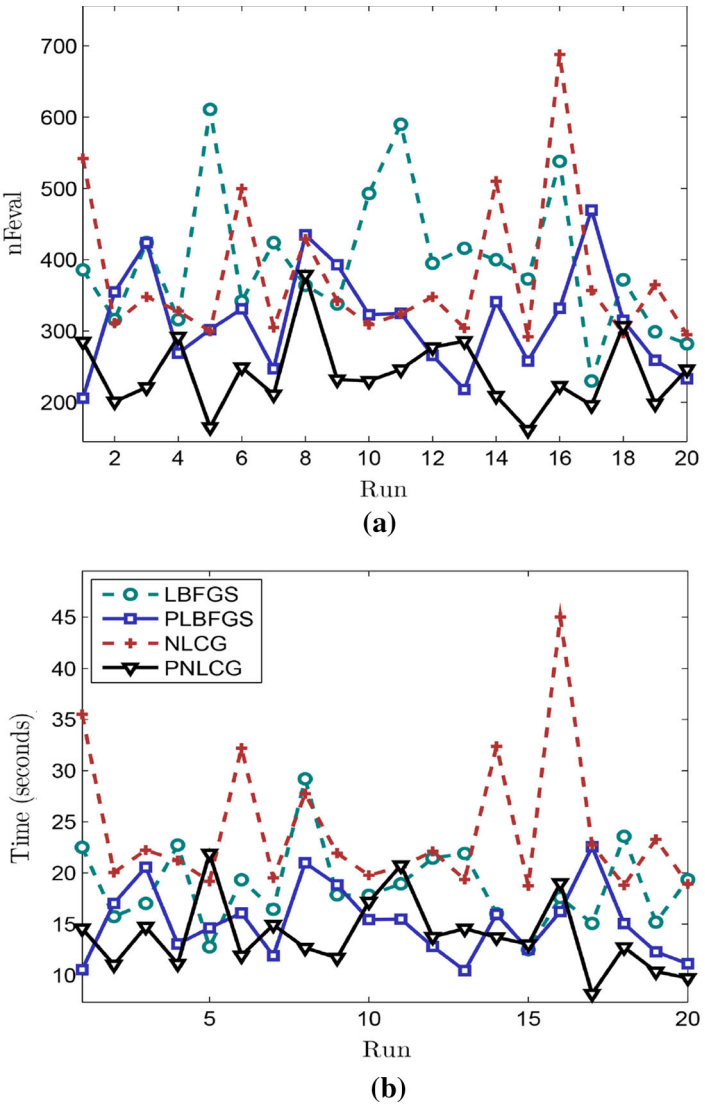

Fig. 6 20 simulations with density function $\rho(x, y) = 1$ on a regular octagon bounded by $[-2, 2] \times [-2, 2]$. **a** Run versus nFeval, **b** run versus time

5.1 Example 3

We first test a constant density function, $\rho(x, y) = 1$, and chose Ω as a regular octagon bounded by $[-2, 2] \times [-2, 2]$ see Fig. 5. For numerical integration we use a second order Gaussian quadrature.

The initial guess affects the performance of all methods tested; see Fig. 6. Over all both the PLBFGS(7) and PNLCG are more stable compared to LBFGS(7) and NLCG. This is the reason for taking averaged results for all tests.

Table 3 Data table for example 3, density function $\rho(x, y) = 1$ on a regular octagon bounded by $[-2, 2] \times [-2, 2]$

Method	Iter	nFeval	Time (s)	$\mathcal{E}$	$\|\mathcal{D}^{-1}\mathcal{F}\|$	$\|\mathcal{F}\|$
Lloyd	1,000	1,000	33.61	1.0378e−2	1.6852e−3	9.4723e−6
LBFGS(7)	373.60	395.45	13.88	1.0362e−2	9.1543e−7	6.8850e−8
PLBFGS(7)	286.75	315.10	15.17	1.0366e−2	9.2603e−7	6.9348e−8
NLCG	363.10	374.70	24.06	1.0365e−2	9.6693e−7	7.2866e−8
PNLCG	219.85	240.70	18.65	1.0366e−2	9.2846e−7	6.9517e−8

Results are comprised of the average from 20 runs

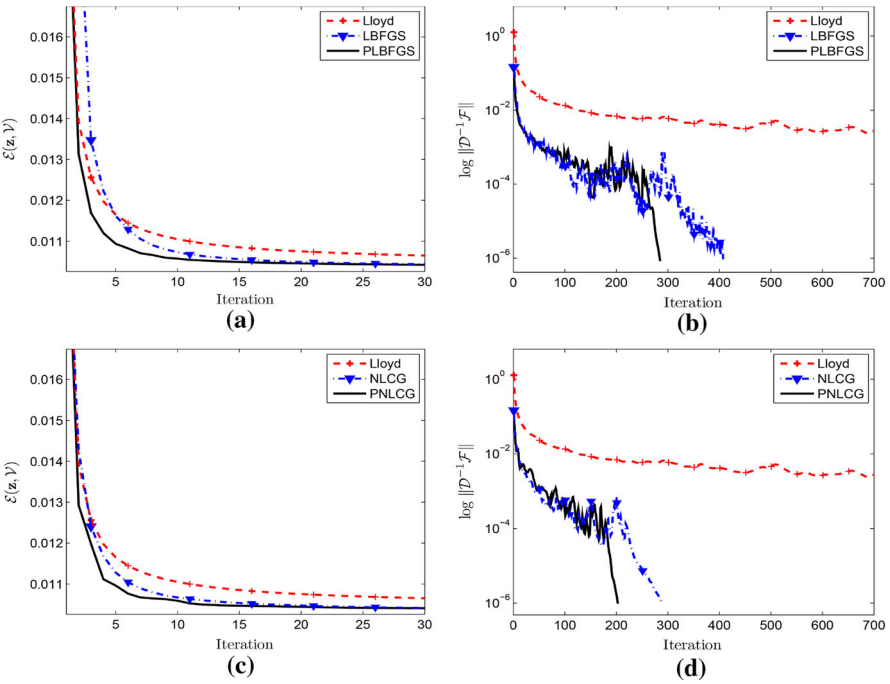

Fig. 7 Energy and error plots for example 3. **a** Lloyd, LBFGS(7) and PLBFGS(7) comparison of iteration versus energy, **b** iteration versus log of l_2-norm weighted error, **c** Lloyd, NLCG and PNLCG comparison of iteration versus energy, **d** iteration versus log of l_2-norm weighted error

From Table 3, we can see both PLBFGS(7) and PNLCG meet the stopping criteria in fewer iterations than their respective counterparts. Since each iteration of preconditioned methods involves an incomplete Chloesky factorization and the drop of iteration steps is not significant, the average total time of twenty runs of PLBFGS(7) is slightly bigger than that of LBFGS(7). For NLCG, we use Hessian to construct the initial guess of line search and thus they are in general more costly than LBFGS despite fewer iterations. Due to the drop of iteration steps, PNLCG is faster than NLCG.

We use a typical run from each optimization method to illustrate the behavior of the energy and error, see Fig. 7. Comparing with Lloyd's method, the energy using BFGS or NLCG drops rapidly, then starts to level off after 10 iterations.

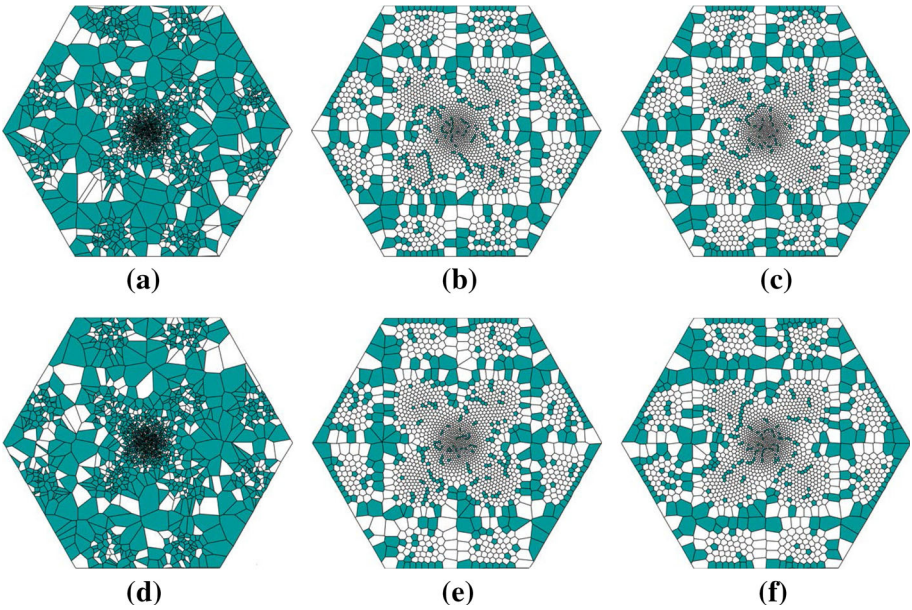

Fig. 8 Voronoi tessellations for example 4. **a** An initial Voronoi tessellation with 2,000 random generators, **b** CVT obtained from a PLBFGS(7) using **a**, **c** CVT obtained from 1,000 iterations of Lloyd's method using **a**, **d** An initial Voronoi tessellation with 2,000 random generators, **e** CVT obtained from a PLBFGS(7) using **d**, **f** CVT obtained from 1,000 iterations of Lloyd's method using **d**

Table 4 Data table for example 4, density function $\rho(x, y) = e^{-20(x^2+y^2)} + \frac{1}{20}\sin^2(\pi x)\ \sin^2(\pi y)$ on a regular hexagon bounded by $[-2, 2] \times [-1.732, 1.732]$

Method	Iter	nFeval	Time (s)	$\mathcal{E}$	$\|\mathcal{D}^{-1}\mathcal{F}\|$	$\|\mathcal{F}\|$
Lloyd	1,000	1,000	50.16	1.7066e−4	3.3112e−3	3.3607e−7
LBFGS(7)	530.80	554.15	28.76	1.7024e−4	9.4883e−7	9.1685e−9
PLBFGS(7)	230.90	252.95	16.37	1.6423e−4	8.9680e−7	1.1374e−8
NLCG	442.25	443.85	39.71	1.7027e−4	9.8116e−7	1.1039e−8
PNLCG	170.90	176.60	18.07	1.6939e−4	9.2365e−7	1.1190e−8

Results are comprised of the average from 20 runs

For constant density, the preconditioned scheme does not seem as much of as improvement. Furthermore, the case with constant density is not challenging as one can start with a regular tessellation using hexagons which is nearly optimal.

5.2 Example 4

For the second 2-D test we take the example 4 in [27]. We use the density $\rho(x, y) = e^{-20(x^2+y^2)} + \frac{1}{20}\sin^2(\pi x)\ \sin^2(\pi y)$ and chose Ω as a regular hexagon bounded by $[-2, 2]\times[-1.732, 1.732]$, see Fig. 8. For numerical integration we use a sixth order Gaussian quadrature which results the time increase of Lloyd method (from 33.61 s in Table 3 to 50.16 s in Table 4).

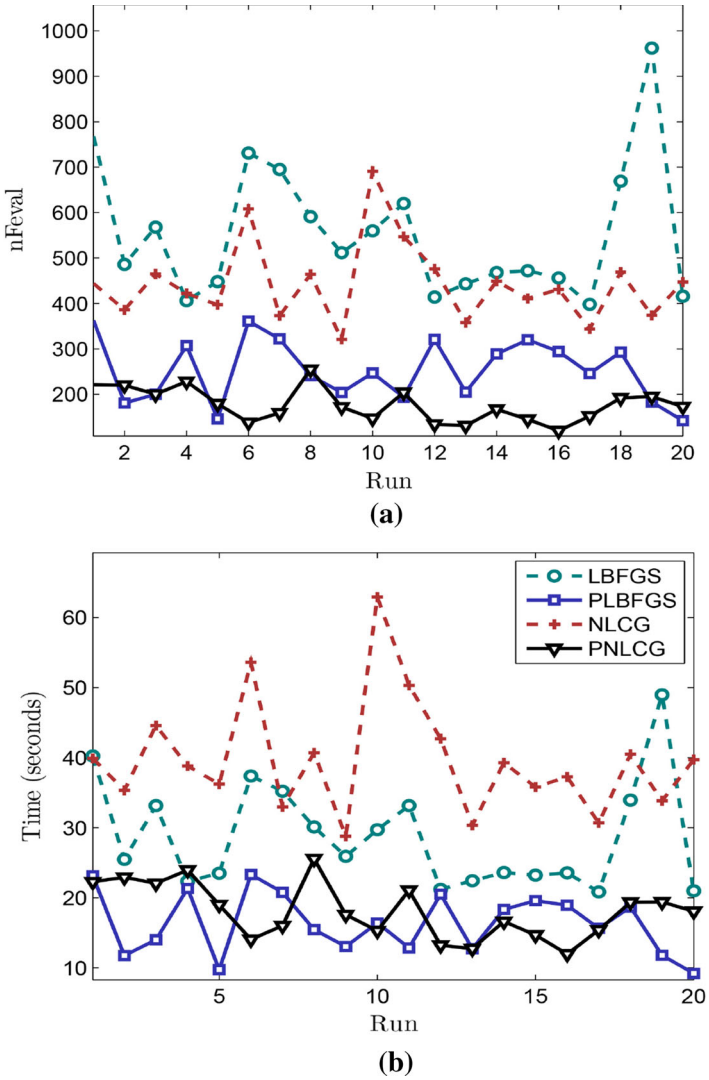

Fig. 9 20 simulations with density function $\rho(x, y) = e^{-20(x^2+y^2)} + \frac{1}{20}\sin^2(\pi x)\ \sin^2(\pi y)$ on a regular hexagon bounded by $[-2, 2] \times [-1.732, 1.732]$. **a** Run versus nFeval, **b** run versus time

In comparing multiple runs, we have both the PLBFGS(7) and PNLCG show more stable than their respective counterparts; see Fig. 9.

For both the LBFGS and NLCG the preconditioned scheme are faster; see Table 4. The reason of the improvement is due to the non-trivial density. In this case, the inverse of graph Laplacian can adjust the density of the generators more efficiently. For the behavior of the energy and error for a typical run see Fig. 10.

To test robustness with respect to the size of generators, we test 2,000 to 8,000 generators with an increment of 1,000. For each size, we use the same stoping criteria of $\|\mathcal{D}^{-1}\mathcal{E}\| \leq$ 1.e−6 which is independent of the size N. With each preconditioned scheme the number of function calls, and hence the number of iterations remains relatively constant; see Fig. 11a.

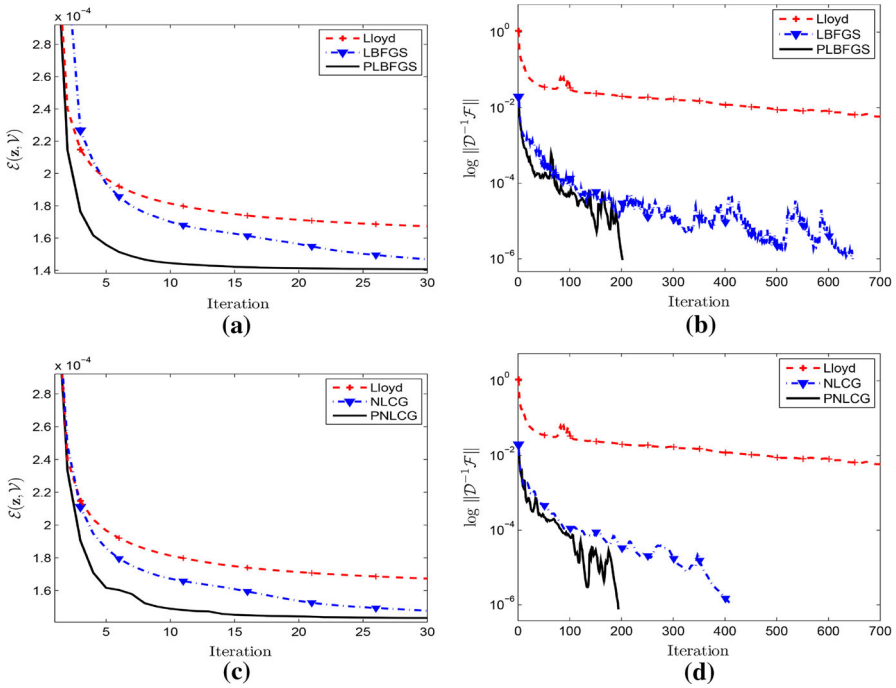

Fig. 10 Energy and error plots for example 4. **a** Lloyd, LBFGS(7) and PLBFGS(7) comparison of iteration versus energy, **b** iteration versus log of l_2-norm weighted error, **c** Lloyd, NLCG and PNLCG comparision of iteration versus energy, **d** iteration versus log of l_2-norm weighted error

So the overall computation complexity is mainly dependent on the complexity of generating Voronoi tessellation, which is $\mathcal{O}(N \log N)$. From Fig. 11b, as the number of generators increases, the time increase is almost linear for both preconditioned schemes. This is in contrast to the nonpreconditioned counter parts.

5.3 Example 5

For the third 2-D test we use the density $\rho(x, y) = e^{-10|x^2+y^2-1|}$ and $\Omega = (-1, 1) \times (-1, 1)$, see Fig. 12. For numerical integration we use a nineth order Gaussian quadrature.

Through multiple runs, the sensitivity to the initial guess results remain consistent with example 4. The initial guess has less effect on both the PLBFGS(7) and PNLCG, as opposed to LBFGS(7) and NLCG; see Fig. 13. The average time needed for the PLBFGS(7) and PNLCG to reach the stopping criteria is roughly the same, about 23–25 s. On average, the LBFGS(7) takes 44 s, while NLCG takes 51 s; see Table 5.

Both the PNLCG and PLBFGS(7) show a similar performance as in Example 4; see Table 5. From Example 3, 4, and 5, we also observe that the performance of PLBFGS and PNLCG is quite robust to different choices of density functions.

The energy for both the PLBFGS(7) and PNLCG level off at about 10 iterations, in half of the iterations for the LBFGS(7) and NLCG schemes to level off. This is illustrated in Fig. 14a, c.

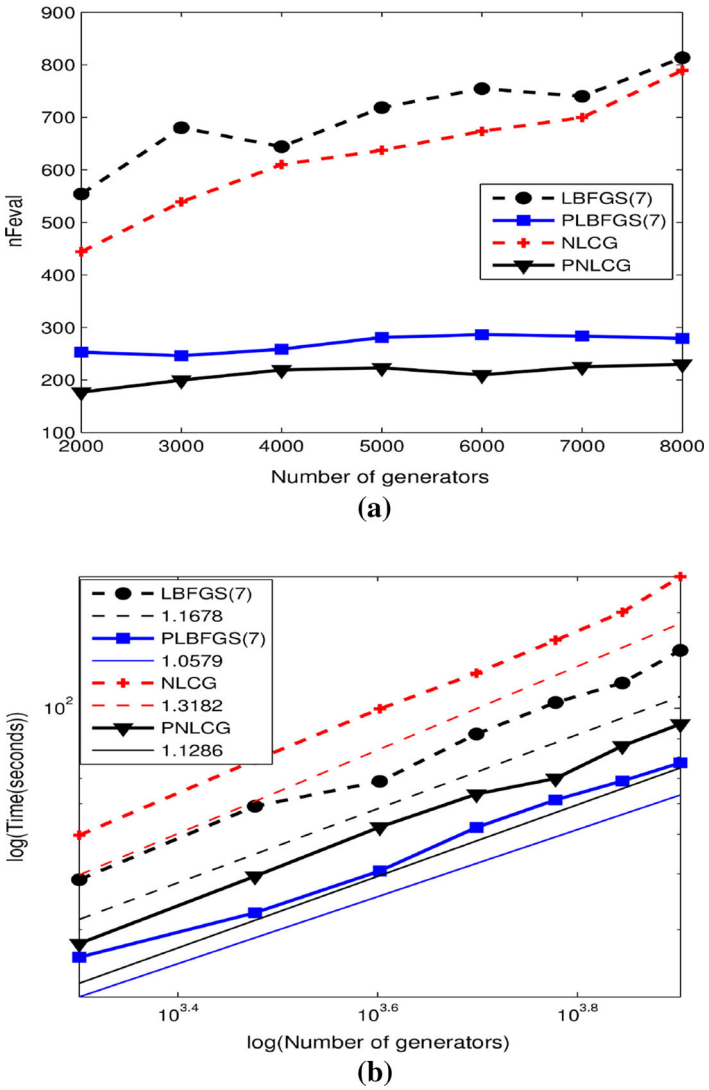

Fig. 11 Figure for robustness test for example 4. The number of generators increase from 2,000 to 8,000 with increment 1,000. **a** Number of generators versus nFeval, **b** log of number of generators versus log of time

Again to display robustness we look at 2,000 to 8,000 generators. The test shows consistent results with example 4. With each preconditioned scheme the number of function calls, and hence the number of iterations remains relatively constant. As the number of generators increases, the time increase is almost linear for both preconditioned schemes. While the nonpreconditioned schemes, on average take about 50 s longer per 1,000 generator increase. The results can be seen in Fig. 15.

The energy values between preconditioned and nonpreconditioned schemes for all tests run are similar. Despite the preconditioned schemes being converging faster, they do not necessarily produce a lower energy value.

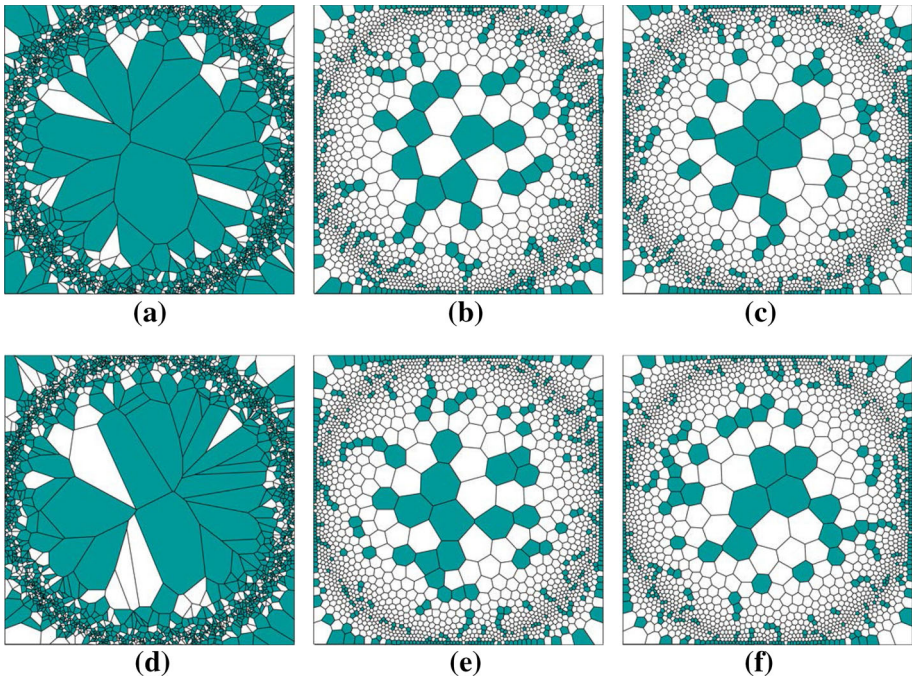

Fig. 12 Voronoi tessellations for example 5. **a** An initial Voronoi tessellation with 2,000 random generators, **b** CVT obtained from a PLBFGS(7) using **a**, **c** CVT obtained from 1,000 iterations of Lloyd's method using **a**, **d** an initial Voronoi tessellation with 2,000 random generators, **e** CVT obtained from a PLBFGS(7) using **d**, **f** CVT obtained from 1,000 iterations of Lloyd's method using **d**

6 Numerical Results: Two-Grid Methods

To further speed up our preconditioned methods and produce a lower energy value, we implement a two-grid method and repeat a non-constant density example in this section. The two-grid algorithm is similar to that presented in Sect. 4 for 1-D CVT computation.

In 2-D, the tricky part is the grid refinement. After the optimization method is executed on the coarse grid, each Voronoi region will contribute roughly the same amount to the energy functional [9,18,24]. A good refinement should preserve this relationship between Voronoi regions. However, since there are no generators on the boundary Γ, with standard uniform refinement the Voronoi regions that intersect the boundary will not preserve the relationship.

A suitable fix is to slightly extend the boundary by some factor α on the coarse grid and after refinement truncate back to the the original domain; see Fig. 16. For a regular polygon with s sides the factor $\alpha = \frac{1}{2N} r \sin\left(\frac{\pi}{s}\right)$ can be used, where N is the number of generators on the coarse grid, and r is the radius of the circle that circumscribes the regular polygon. We start with a triangulation of each Voronoi regions in a slightly extended domain with the factor α. Then we concatenate the midpoints of the edges of each Voronoi region onto the generators and truncate back to the original domain. Refining in such approximately quadruples the number of generators and preserves the similar energy values of each Voronoi region, see Fig. 16. Figure 17b is an example of a fine grid right after refinement from Fig. 17a. This is an evidence of this refinement being good as the energy decreased to slightly more than one

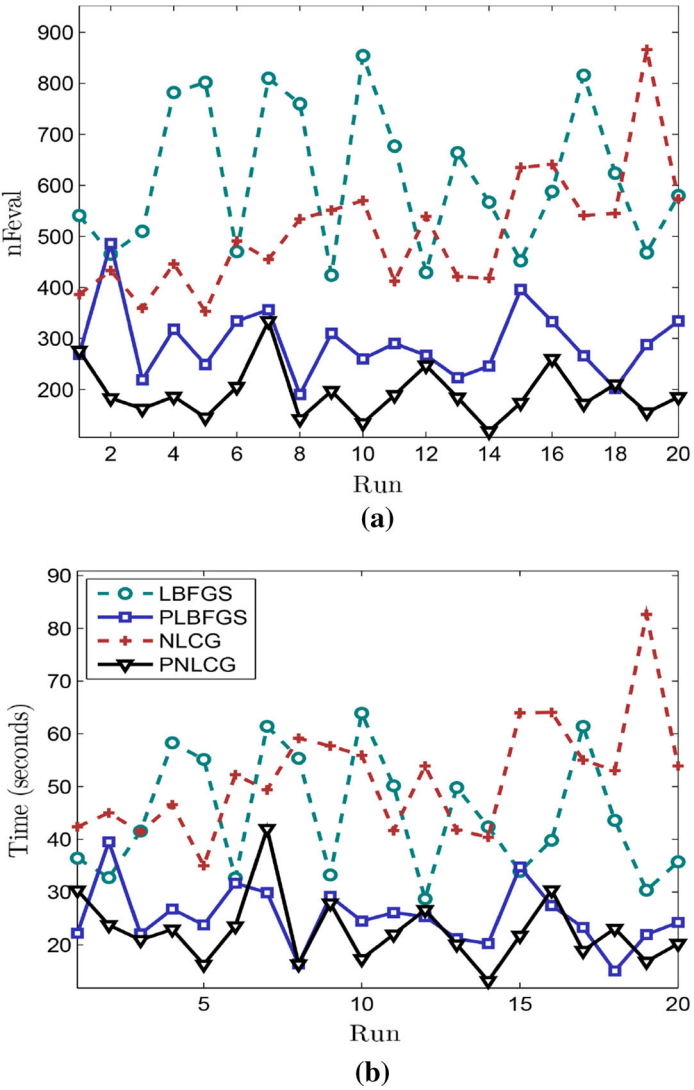

Fig. 13 20 simulations with density function $\rho(x, y) = e^{-10|x^2+y^2-1|}$ and $\Omega = (-1, 1) \times (-1, 1)$. **a** Run versus nFeval, **b** run versus time

forth of the old one when number of generators quadruples, i.e. the energy decays at nearly the optimal rate of N^{-1}.

Starting with 2,000 generators on the coarse grid, we set the error tolerance of 1.e−4, i.e. $\|\mathcal{D}^{-1}\mathcal{F}(\mathbf{z})\| < 1.e{-}4$. We use the same optimization method for the coarse grid as we do on the fine grid. We use Lloyd's method as a reference, and compare Lloyd's method, PLBFGS(7) and PNLCG, with and without the two-grid method. Just as in Sect. 5, we average the results from 20 runs. For each run we use the same number of generators for the single grid and two-grid methods, i.e., if one simulation refines to 7,900 generators for the two grid method, we run a simulation with 7,900 generators for a single grid method. This way we

Table 5 Data table for example 5, density function $\rho(x, y) = e^{-10|x^2+y^2-1|}$ and $\Omega = (-1, 1) \times (-1, 1)$

Method	Iter	nFeval	Time (s)	$\mathcal{E}$	$\|\mathcal{D}^{-1}\mathcal{F}\|$	$\|\mathcal{F}\|$
Lloyd	1,000	1,000	69.10	7.4696e−5	9.6136e−4	1.3990e−7
LBFGS(7)	579.45	614.15	44.33	7.4559e−5	9.7465e−7	8.3022e−9
PLBFGS(7)	245.75	291.80	25.27	7.4572e−5	9.1869e−7	1.5361e−8
NLCG	504.70	508.45	51.75	7.4565e−5	9.7608e−7	1.0002e−8
PNLCG	178.35	192.90	22.71	7.4548e−5	9.1381e−7	1.6176e−8

Results are comprised of the average from 20 runs

directly compare energy values for each simulation; see Fig. 19. For Fig. 18 we use a typical run to illustrate the energy and error decay.

We return to the same density function as in example 4 in Sect. 5: $\rho(x, y) = e^{-20(x^2+y^2)} + \frac{1}{20}\sin^2(\pi x)\sin^2(\pi y)$ on regular hexagon bounded by $[-2, 2] \times [-1.732, 1.732]$. We have tested other examples and the performance is similar.

Both the PLBFGS(7) and PNLCG, only take about 5 iterations using the two grid method before the energy levels off. Using the PLBFGS(7) scheme directly starting from the fine grid takes about 12 iterations. Using the PNLCG starting from the same number of generators takes 15 iterations before the energy levels off; see Fig. 18a, c.

There is no advantage for using a two-grid method for Lloyd's method. On each level the maximum number of iterations allowed is reach before the stopping criteria is met. Looking

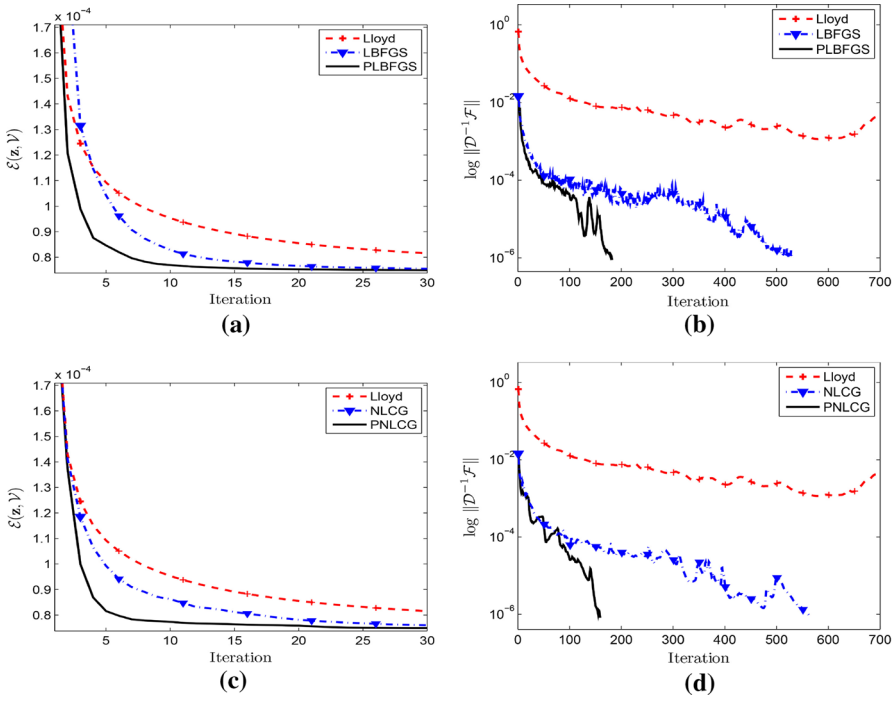

Fig. 14 Energy and error plots for Example 3. **a** Lloyd, LBFGS(7) and PLBFGS(7) comparison of iteration versus energy. **b** Iteration versus log of weighted error. **c** Lloyd, NLCG and PNLCG comparision of iteration versus energy. **d** Iteration versus log of weighted error

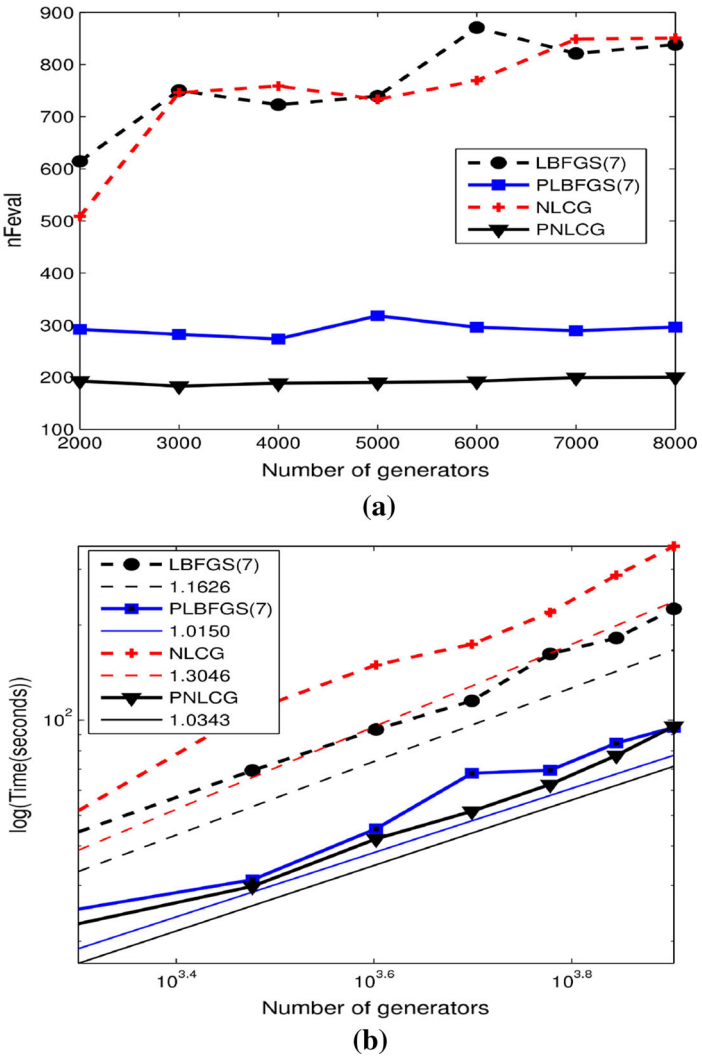

Fig. 15 Figure for robustness test for example 4. The number of generators increase from 2,000 to 8,000 with increment 1,000. **a** Number of generators versus nFeval. **b** Log of number of generators versus log of time

at Lloyd's method, from the single grid to the two-grid method there is only a slight drop in the energy. This is in contrast to the preconditioned schemes. The PLBFGS(7) takes, on average, 285 iterations with a single grid and 260 using a two-grid method. However for the two-grid method, on average, 47 of these iterations are done on the coarse grid. Similarly, on average, PNLCG using the two-grid takes 238 total iterations, 15 more iterations than the one-grid. Yet again, on average, 69 of these iterations are done on the coarse grid. Iterations on the coarse grid are approximately three and a half times computationally cheaper than iterations on the fine grid. In other words, 60 iterations on the coarse grid amounts to approximately 17 iterations on the fine grid.

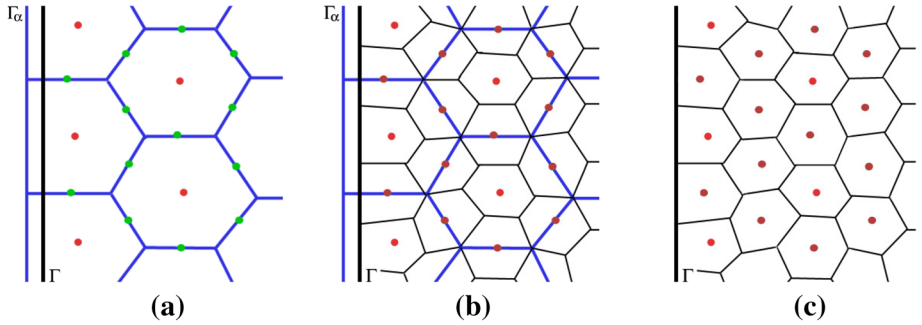

Fig. 16 Refinement of Voronoi Tessellation. **a** Voronoi Tessellation on the coarse grid. Green nodes are the midpoints of the interior cell edges, Γ_α is the extend boundary, Γ is the orignal boundary. **b** Coarse grid and refined grid over-layed. **c** Refined grid truncated back to original domain

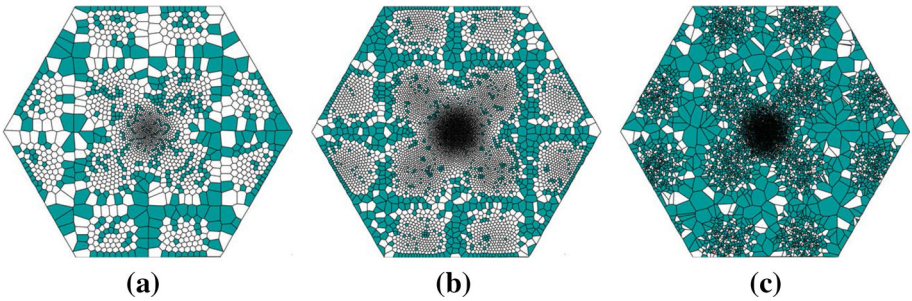

Fig. 17 Voronoi tessellation refinement for example 6. **a** Voronoi tessellation with 2,000 generators after smoothing criteria of $\|\mathcal{D}^{-1}\mathcal{F}\| < 1.e{-}4$ has been met, $\mathcal{E} = 1.5284e{-}4$, **b** refined Voronoi tessellation from 2,000 to 7,902 generators without smoothing on fine grid, $\mathcal{E} = 4.5757e{-}5$, **c** Voronoi tessellation with 7,902 random generators, $\mathcal{E} = 9.9048e{-}5$

For the test we display, using the two-grid method with both the PLBFGS(7) and PNLCG, we have about a 24 % decrease in time versus the same stopping criteria without using the two-grid method. The results can be see in Table 6. These simulations show that not only does a two-grid method speed up convergence, on average, it also helps in finding a lower energy; see Fig. 19.

Although we only show results for a two-grid method, grid cascading, with a suitable refining method, will further improve the performance for applications needing very fine meshes.

7 Summary and Further Works

We have presented an efficient preconditioner and a two-grid method for classical optimization methods on finding a stable CVT. All optimization methods included in produce a stable CVT, see discussion in Sect. 5.2 in [27]. Comparing with Lloyd's method and NLCG and LBFGS methods without preconditioner, we have a much more efficient method for finding a stable CVT. Although we only show 1D and 2D results, our approach is easily adapted for meshes on surfaces and in 3D. For surface meshes, the graph Laplacian construction is the

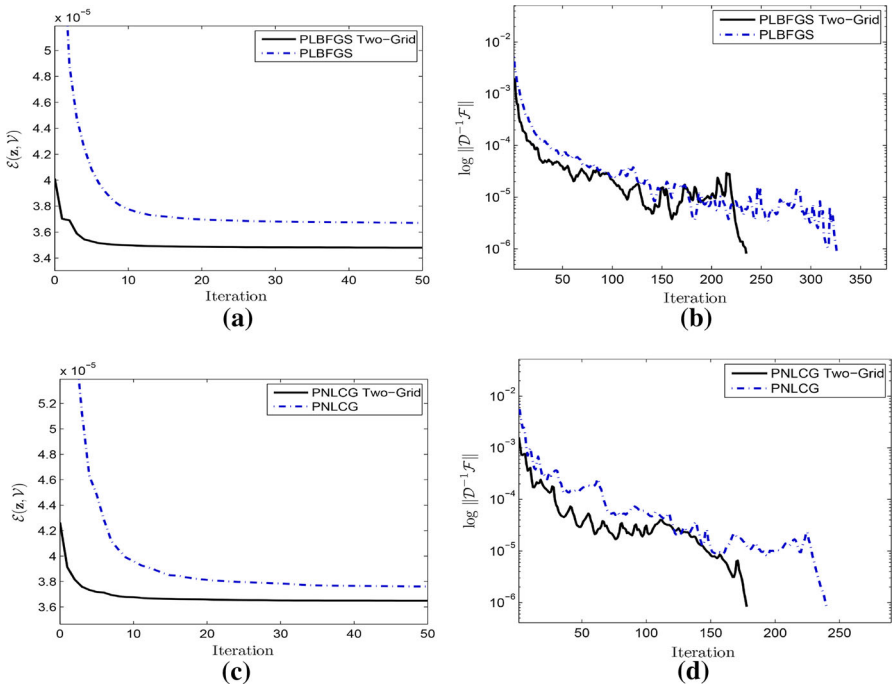

Fig. 18 Two grid method and single grid comparision of energy and weighted error. **a** PLBFGS(7) with 7,894 generators on the fine grid, iteration versus energy, **b** iteration versus log of l_2-norm of weighted error, **c** PNLCG with 7,897 generators on the fine grid, iteration versus energy, **d** iteration versus log of l_2-norm weighted error

Table 6 Data table for Example 6

Method	Grid	Level	nFeval	Time (s)	$\mathcal{E}$	$\|\mathcal{D}^{-1}\mathcal{F}\|$
Lloyd	Two	Coarse	1,000	54.47	9.9331e−5	2.1608e−3
		Fine	1,000	275.26	3.8728e−5	9.2573e−5
	Single	–	1,000	274.89	3.9421e−5	4.6894e−4
PLBFGS(7)	Two	Coarse	46.55	3.12	1.4347e−4	9.0291e−5
		Fine	213.45	55.20	3.4767e−5	9.8198e−7
	Single	–	285.65	76.85	3.6781e−5	8.0917e−7
PNLCG	Two	Coarse	68.85	6.74	4.2861e−5	8.2985e−5
		Fine	168.60	47.89	3.6463e−5	9.5912e−7
	Single	–	252.25	71.88	3.7439e−5	9.6393e−7

Comparison for Lloyd's method, PLBFGS(7) and PNLCG with a single grid and two-grid method. Results are averaged from 20 runs with a coarse grid starting at 2,000 generators. Average number of generators on the fine grid is 7,895

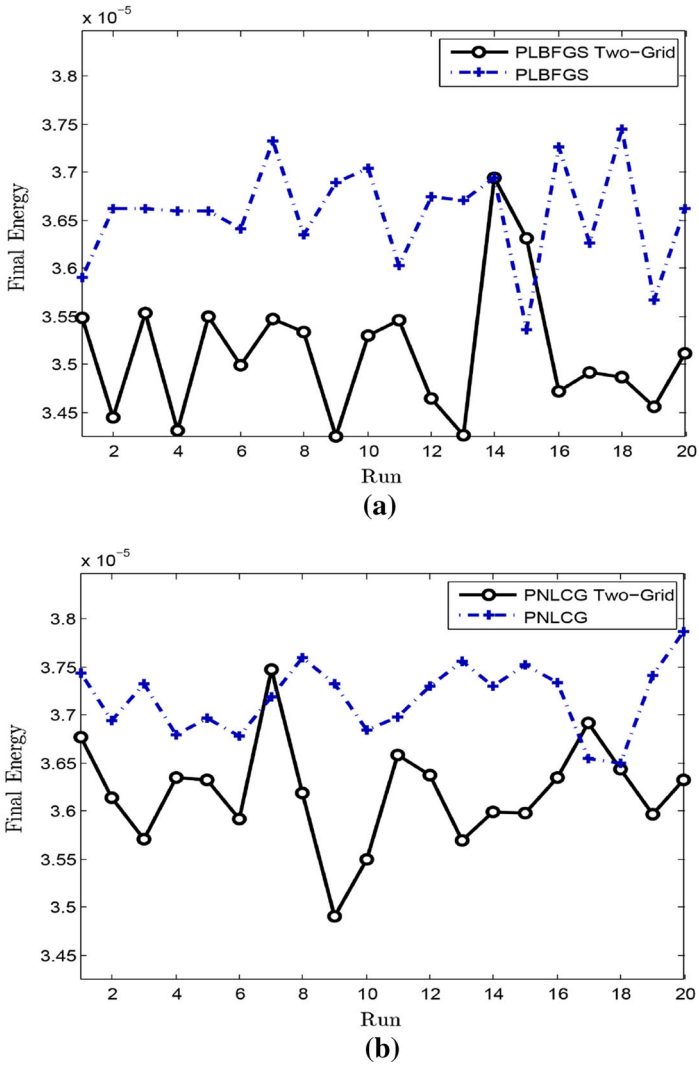

Fig. 19 Comparison of final energy values for 20 simulations for two and single grid methods. **a** PLBFGS, **b** PNLCG

same as 2D. For 3D, the Voronoi tessellation can be decomposed into a tetrahedral mesh, and the graph Laplacian is constructed through tetrahedrons.

We plan to apply our fast methods for mesh generation and optimization [10,30,34], image processing and computer graphics [11,12], and other applications [9].

References

1. Benzi, M., Tůma, M.: A comparative study of sparse approximate inverse preconditioners. Appl. Numer. Math. **30**, 305–340 (1999)
2. Brandt, A.: Algebraic multigrid theory: the symmetric case. Appl. Math. Comput. **19**(1–4), 23–56 (1986)

3. Chen, L.: *i*FEM: An Integrated Finite Element Methods Package in MATLAB. Technical Report, UC Irvine
4. Chen, L., Holst, M.: Efficient mesh optimization schemes based on optimal Delaunay triangulations. Comput. Methods Appl. Mech. Eng. **200**, 967–984 (2011)
5. Chen, L., Xu, J.: Optimal Delaunay triangulations. J. Comput. Math. **22**(2), 299–308 (2004)
6. Du, Q., Emelianenko, M.: Acceleration schemes for computing centroidal Voronoi tessellations. Numer. Linear Algebra **13**(2–3), 173–192 (2006)
7. Du, Q., Emelianenko, M.: Uniform convergence of a nonlinear energy-based multilevel quantization scheme. SIAM J. Numer. Anal. **46**(3), 1483–1502 (2008)
8. Du, Q., Emelianenko, M., Ju, L.: Convergence of the Lloyd algorithm for computing centroidal Voronoi tessellations. SIAM J. Numer. Anal. **44**(1), 102–119 (2006)
9. Du, Q., Faber, V., Gunzburger, M.: Centroidal voronoi tessellations: applications and algorithms. SIAM Rev. **41**(4), 637–676 (1999)
10. Du, Q., Gunzburger, M.: Grid generation and optimization based on centroidal Voronoi tessellations. Appl. Math. Comput. **133**, 591–607 (2002)
11. Du, Q., Gunzburger, M., Ju, L.: Advances in studies and applications of centroidal Voronoi tessellations. Numer. Math. Theor. Methods Appl. **3**(2), 119–142 (2010)
12. Du, Q., Gunzburger, M., Ju, L., Wang, X.: Centroidal Voronoi tessellation algorithms for image compression, segmentation, and multichannel restoration. J. Math. Imaging Vis. **24**, 177–194 (2006)
13. Emelianenko, M.: Fast multilevel CVT-based adaptive data visualization algorithm. Numer. Math. Theor. Methods Appl. **3**, 195–211 (2010)
14. Emelianenko, M., Ju, L., Rand, A.: Nondegeneracy and weak global convergence of the Lloyd algorithm in $\mathbb{R}^d$. SIAM J. Numer. Anal. **46**(3), 1423–1441 (2008)
15. Fabri, A., Giezeman, G., Kettner, L., Schirra, S., Schönherr, S., et al.: On the design of CGAL, the computational geometry algorithms library (1998)
16. Fejes Tóth, G.: A stability criterion to the moment theorem. Stud. Sci. Math. Hung. **38**(1), 209–224 (2001)
17. Fletcher, R., Reeves, C.M.: Function minimization by conjugate gradients. Comput. J. **7**(2), 149–154 (1964)
18. Gersho, A.: Asymptotically optimal block quantization. IEEE Trans. Inf. Theory **25**(4), 373–380 (1979)
19. Gray, R.M., Neuhoff, D.L.: Quantization. IEEE Trans. Inf. Theory **44**, 2325–2384 (1998)
20. Gruber, P.M.: Optimum quantization and its applications. Adv. Math. **186**(2), 456–497 (2004)
21. Hager, W.W., Zhang, H.: A survey of nonlinear conjugate gradient methods. Pac. J. Optim. **2**(1), 35–58 (2006)
22. Hestenes, M.R., Stiefel, E.: Methods of Conjugate Gradients for Solving Linear Systems (1952)
23. Koren, Y., Yavneh, I.: Adaptive multiscale redistribution for vector quantization. SIAM J. Sci. Comput. **27**(5), 1573–1593 (2006)
24. Koren, Y., Yavneh, I., Spira, A.: A multigrid approach to the scalar quantization problem. IEEE Trans. Inform. Theory **51**(8), 2993–2998 (2005)
25. Lin, C., Moré, J.J.: Incomplete cholesky factorizations with limited memory. SIAM J. Sci. Comput. **21**(1), 24–45 (1999)
26. Liu, D.C., Nocedal, J.: On the limited memory BFGS method for large scale optimization. Math. Program. **45**, 503–528 (1989)
27. Liu, Y., Wang, W., Lévy, B., Sun, F., Yan, D., Lu, L., Yang, C.: On centroidal Voronoi tessellation—energy smoothness and fast computation. ACM Trans. Graph. **28**, 101:1–101:17 (2009)
28. Livne, O., Brandt, A.: Lean algebraic multigrid (LAMG): fast graph Laplacian linear solver. ArXiv e-prints (2011)
29. Lloyd, S.: Least squares quantization in PCM. IEEE Trans. Inform. Theory **28**(2), 129–137 (1982)
30. Nguyen, H., Burkardt, J., Gunzburger, M., Ju, L., Saka, Y.: Constrained CVT meshes and a comparison of triangular mesh generators. Comput. Geom. **42**(1), 1–19 (2009)
31. Nocedal, J., Wright, S.J.: Numerical Optimization. Springer, Berlin (1999)
32. Polak, E., Ribiere, G.: Note sur la convergence de méthodes de directions conjuguées. Revue. Fr. Inf. Rech. Oper. **3**(1), 35–43 (1969)
33. Ruge, J.W., Stüben, K.: Algebraic multigrid (AMG). In: McCormick, S.F. (ed.) Multigrid Methods, Frontiers in Applied Mathematics, vol. 3, pp. 73–130. SIAM, Philadelphia, PA (1987)
34. Wang, D., Du, Q.: Mesh optimization based on the centroidal voronoi tessellation. Int. J. Numer. Anal. Model. **2**, 100–113 (2005)
35. Xu, J.: Two-grid discretization techniques for linear and nonlinear PDEs. SIAM J. Numer. Anal. **33**(5), 1759–1777 (1996)
36. Xu, J.: Fast Poisson-based solvers for linear and nonlinear PDEs. In: Proceedings of the International Congress of Mathematics, vol. 4, pp. 2886–2912. (2010)

37. Zhang, J., Emelianenko, M., Du, Q.: Periodic centroidal Voronoi tessellations. IJNAM **9**(4), 950–969 (2012)
38. Zimmer, H.: Voronoi and Delaunay techniques. In: Proceedings of Lecture Notes, Computer Sciences, vol. 8. (2005)