Fast Computation for Centroidal Voronoi Tessellations

James Hateley, Huayi Wei, Long Chen

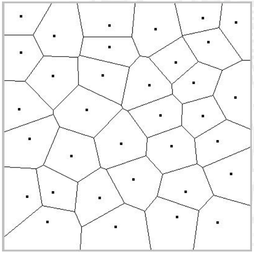
Department of Mathematics
University of California, Irvine

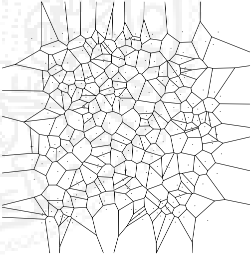
Outline

Introduction

A *Voronoi Tesselation* (Voronoi Diagram) $\mathcal{V} = \{V_i\}_{i=1}^N$ is a special type of partitioning of an open subset Ω of $\mathbb{R}^n$. This partitioning of Ω is determined by distances to a specified set of generators $\mathbf{z} = \{z_i\}_{i=1}^N$. Each *Voronoi region* V_i will satisfy;

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



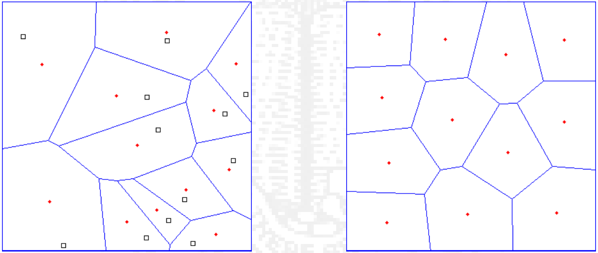


Introduction

A *Centroidal Voronoi Tessellation* (CVT) is a Voronoi Tessellation in where the generators correspond to the centroids of each region.

Introduction

A *Centroidal Voronoi Tessellation* (CVT) is a Voronoi Tessellation in where the generators correspond to the centroids of each region.



Left: (Squares) Voronoi Tessellation on a square, (dots) CVT.

Right: Stable CVT on the same region.

Energy Minimization

A CVT is also defined to be a critical point of the mean square distortion measure (variance),

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

Where $\rho(x)$ is a given density function.

- A *stable CVT* corresponds to a local minimizer of $\mathcal{E}(\mathbf{z}, \mathcal{V})$.
- An *optimal CVT* corresponds to the global minimizer.
- An *unstable CVT* corresponds to a saddle point.

Application

There is a wide range of applications for CVTs:

Application

There is a wide range of applications for CVTs:

- Computer graphics

Application

There is a wide range of applications for CVTs:

- Computer graphics
- Data compression

Application

There is a wide range of applications for CVTs:

- Computer graphics
- Data compression
- Mesh generation

Application

There is a wide range of applications for CVTs:

- Computer graphics
- Data compression
- Mesh generation
- Optimal quantization

Application

There is a wide range of applications for CVTs:

- Computer graphics
- Data compression
- Mesh generation
- Optimal quantization

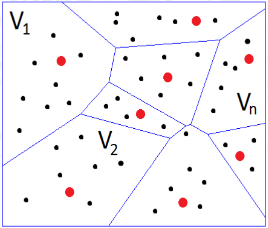
Du, Qiang and Faber, Vance and Gunzburger, Max. Centroidal Voronoi Tessellations: Applications and Algorithms. SIAM Rev.. 41(4):637-676, 1999.

Application

There is a wide range of applications for CVTs:

- Computer graphics
- Data compression
- Mesh generation
- Optimal quantization

Du, Qiang and Faber, Vance and Gunzburger, Max. Centroidal Voronoi Tessellations: Applications and Algorithms. SIAM Rev.. 41(4):637-676, 1999.



Main Idea: Graph Laplacian Precondtiioner

We propose using a weighted graph Laplacian as a preconditioner in a quasi-Newton scheme for finding a stable CVT. There are several benefits for choosing the graph Laplacian instead of the Hessian.

Main Idea: Graph Laplacian Precondtiioner

We propose using a weighted graph Laplacian as a preconditioner in a quasi-Newton scheme for finding a stable CVT. There are several benefits for choosing the graph Laplacian instead of the Hessian.

- The graph Laplacian is easy to assemble.

Main Idea: Graph Laplacian Precondtiioner

We propose using a weighted graph Laplacian as a preconditioner in a quasi-Newton scheme for finding a stable CVT. There are several benefits for choosing the graph Laplacian instead of the Hessian.

- The graph Laplacian is easy to assemble.
- Captures the essential features of the Hessian.

Main Idea: Graph Laplacian Precondtiioner

We propose using a weighted graph Laplacian as a preconditioner in a quasi-Newton scheme for finding a stable CVT. There are several benefits for choosing the graph Laplacian instead of the Hessian.

- The graph Laplacian is easy to assemble.
- Captures the essential features of the Hessian.
- The inverse of the graph Laplacian can be computed efficiently (i.e. AMG)

Lloyd's Method

Two step gradient method:

- Fix $\mathbf{z}$, optimize partition

Lloyd's Method

Two step gradient method:

- Fix $\mathbf{z}$, optimize partition
- Fix $\mathcal{V}$, optimize $\mathbf{z}$

Lloyd's Method

Two step gradient method:

- Fix $\mathbf{z}$, optimize partition
- Fix $\mathcal{V}$, optimize $\mathbf{z}$

Lloyd Iteration

1. Construct Voronoi diagram $\mathcal{V}(\mathbf{z}_k)$
2. Update $z_{i,k+1} = \left(\int_{V_i} \rho(x) \; dx \right)^{-1} \int_{V_i} x\rho(x) \; dx$

Lloyd's Method

Two step gradient method:

- Fix $\mathbf{z}$, optimize partition
- Fix $\mathcal{V}$, optimize $\mathbf{z}$

Lloyd Iteration

1. Construct Voronoi diagram $\mathcal{V}(\mathbf{z}_k)$
2. Update $z_{i,k+1} = \left(\int_{V_i} \rho(x)\ dx \right)^{-1} \int_{V_i} x\rho(x)\ dx$

Lloyd's method is robust with a convergence rate of $1 - \mathcal{O}(h^2)$, where $h = \min_i \operatorname{diam}(V_i)$. Hence the larger the size of the problem, the slower the rate of convergence.

Two Geometrical Multilevel Methods

There are two approaches to optimizing this energy:

- FAS: Brandt, Yavneh.
- Subspace Optimization Method: Tai and Xu, Du and Emelianenko.

Two Geometrical Multilevel Methods

There are two approaches to optimizing this energy:

- FAS: Brandt, Yavneh.
- Subspace Optimization Method: Tai and Xu, Du and Emelianenko.

Difficult to implement in high dimensions.

Two Geometrical Multilevel Methods

There are two approaches to optimizing this energy:

- FAS: Brandt, Yavneh.
- Subspace Optimization Method: Tai and Xu, Du and Emelianenko.

Difficult to implement in high dimensions.

Our approach: classic optimization methods with a good preconditioned.

Liu, Y. and Wang, W. and Lévy, B. and Sun, F. and Yan, D.M. and Lu, L. and Yang, C.. On centroidal voronoi tessellation – energy smoothness and fast computation. ACM Transactions on Graphics (TOG). 28(4):101, 2009.

Classic optimization methods using LU decomposition of Hessian matrix.

A formula for $\mathcal{H}$:

$$\begin{cases} \dfrac{\partial^2 F}{\partial x_{ik}^2} = 2m_i - \sum_{j\in \mathrm{J}_i} \int_{\Omega_i \cap \Omega_j} \dfrac{2}{\|\mathbf{x}_j - \mathbf{x}_i\|}(x_{ik} - x_k)^2 \rho(\mathbf{x})\, \mathrm{d}\sigma, \\ \dfrac{\partial^2 F}{\partial x_{ik}\partial x_{i\ell}} = -\sum_{j\in \mathrm{J}_i} \int_{\Omega_i \cap \Omega_j} \dfrac{2}{\|\mathbf{x}_j - \mathbf{x}_i\|}(x_{ik} - x_k)(x_{i\ell} - x_\ell) \rho(\mathbf{x})\, \mathrm{d}\sigma, & k \neq \ell, \\ \dfrac{\partial^2 F}{\partial x_{ik}\partial x_{j\ell}} = \int_{\Omega_i \cap \Omega_j} \dfrac{2}{\|\mathbf{x}_j - \mathbf{x}_i\|}(x_{ik} - x_k)(x_{j\ell} - x_\ell) \rho(\mathbf{x})\, \mathrm{d}\sigma, & j \in \mathrm{J}_i, \\ \dfrac{\partial^2 F}{\partial x_{ik}\partial x_{j\ell}} = 0, & j \neq i,\ j \notin \mathrm{J}_i. \end{cases}$$

Nonlinear Average

A Voronoi tessellation $\mathcal{V}$ is a CVT if

$$z_i = \left(\int_{V_i} \rho(x) \; dx \right)^{-1} \int_{V_i} x\rho(x) \; dx, x \in \mathbb{R}^n \tag{2}$$

Nonlinear Average

A Voronoi tessellation $\mathcal{V}$ is a CVT if

$$z_i = \left(\int_{V_i} \rho(x)\ dx \right)^{-1} \int_{V_i} x\rho(x)\ dx, x \in \mathbb{R}^n \tag{2}$$

We can view z_i as a nonlinear average of its neighbors.

$$z_i = \sum_{j \in \mathcal{J}_i} w_j z_j \tag{3}$$

Where $\mathcal{J}_i$ are the neighboring Voronoi regions of $\mathcal{V}_i$.

Nonlinear Average

A Voronoi tessellation $\mathcal{V}$ is a CVT if

$$z_i = \left(\int_{V_i} \rho(x)\ dx \right)^{-1} \int_{V_i} x\rho(x)\ dx, x \in \mathbb{R}^n \tag{2}$$

We can view z_i as a nonlinear average of its neighbors.

$$z_i = \sum_{j \in \mathcal{J}_i} w_j z_j \tag{3}$$

Where $\mathcal{J}_i$ are the neighboring Voronoi regions of $\mathcal{V}_i$.
The idea is to approximate as a linear average.

Nonlinear Average

A Voronoi tessellation $\mathcal{V}$ is a CVT if

$$z_i = \left(\int_{V_i} \rho(x)\ dx \right)^{-1} \int_{V_i} x\rho(x)\ dx, x \in \mathbb{R}^n \qquad (2)$$

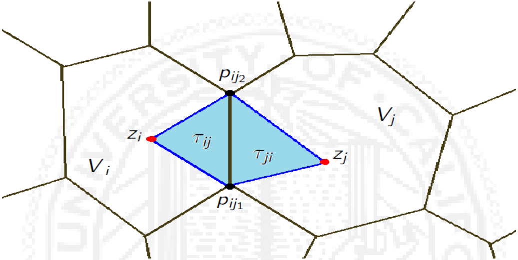
We can view z_i as a nonlinear average of its neighbors.

$$z_i = \sum_{j \in \mathcal{J}_i} w_j z_j \qquad (3)$$

Where $\mathcal{J}_i$ are the neighboring Voronoi regions of $\mathcal{V}_i$. The idea is to approximate as a linear average.

$$a_{ii} z_i = \sum_{j \in \mathcal{J}_i} a_{ij} z_j \qquad (4)$$

Graph Laplacian Construction



$e_{ij} = \overline{V}_i \cap \overline{V_j}$, the edge of two neighboring Voronoi regions V_i, and V_j.

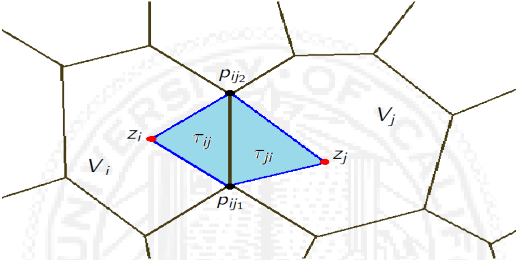
p_{ij_1} and p_{ij_2} as the end points of e_{ij}.

Keeping positive orientation denote the element

$$\tau_{ij} = [z_i, p_{ij_1}, p_{ij_2}], \text{ and } \tau_{ji} = [z_j, p_{ij_2}, p_{ij_1}].$$

Fast Computation for CVTs

Graph Laplacian Construction



$$A = \begin{cases} a_{ij} = -\displaystyle\int_{\tau_{ij}\cup\tau_{ji}} \rho(x)\ dx & \text{if } j \in \mathcal{J}_i, \partial\Omega \cap \partial V_i = \emptyset \\ a_{ij} = -2\displaystyle\int_{\tau_{ij}} \rho(x)\ dx & \text{if } j \in \mathcal{J}_i, \partial\Omega \cap \partial V_i \neq \emptyset \\ a_{ii} = \displaystyle\sum_{j\in\mathcal{J}_i} a_{ij} & \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Graph Laplacian comparison versus $\mathcal{H}$

A direct comparison of $\mathcal{H}$ and our graph-Laplacian A shows why it is such a convenient choice for a preconditioner.

Graph Laplacian comparison versus $\mathcal{H}$

A direct comparison of $\mathcal{H}$ and our graph-Laplacian A shows why it is such a convenient choice for a preconditioner.

graph-Laplacian	Hessian
Symmetric M-matrix	Symmetric but not necessarily definite
Efficient to compute	costly to construct
optimal solver	requres Modified Cholesky decomposition

Quasi-Newton

Given an approximation of the Hessian B

Newton-Type Iterations

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

Where α satisfys the Wolfe conditions.

Quasi-Newton

Given an approximation of the Hessian B

Newton-Type Iterations

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

Where α satisfys the Wolfe conditions.

- If B is $\mathcal{H}$, then we have Newton's method.

Quasi-Newton

Given an approximation of the Hessian B

Newton-Type Iterations

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

Where α satisfys the Wolfe conditions.

- If B is $\mathcal{H}$, then we have Newton's method.
- If $B = A$ then we have a quasi-Newton method.

Quasi-Newton

Given an approximation of the Hessian B

Newton-Type Iterations

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

Where α satisfys the Wolfe conditions.

- If B is $\mathcal{H}$, then we have Newton's method.
- If $B = A$ then we have a quasi-Newton method.
- If $B = \text{diag}(A)$ then we have a quasi-Newton method which preforms similar to Lloyd's method.

Preconditioned Nonlinear Conjugate Gradient (P-NLCG)

After initializing $\beta_0 = -A^{-1}\mathcal{F}(\mathbf{z}_0)$

NLCG Iteration

1. Calculate β_k
2. Update conjugate direction $p_k = -\nabla\mathcal{E}(\mathbf{z}_k) + \beta_k p_{k-1}$
3. Update with line search $\mathbf{z}_{k+1} = \mathbf{z}_k + \alpha_k p_k$

Preconditioned Nonlinear Conjugate Gradient (P-NLCG)

After initializing $\beta_0 = -A^{-1}\mathcal{F}(\mathbf{z}_0)$

NLCG Iteration

1. Calculate β_k
2. Update conjugate direction $p_k = -\nabla\mathcal{E}(\mathbf{z}_k) + \beta_k p_{k-1}$
3. Update with line search $\mathbf{z}_{k+1} = \mathbf{z}_k + \alpha_k p_k$

We choose to impliment Polak-Ribière update.

$$\beta_k^{PR} = \frac{\mathcal{F}(\mathbf{z}_k)^\top \left[\mathcal{F}(\mathbf{z}_k) - \mathcal{F}(\mathbf{z}_{k-1})\right]}{\mathcal{F}(\mathbf{z}_k)^\top \mathcal{F}(\mathbf{z}_{k-1})}$$

Preconditioned Limited Memory BFGS (P-L-BFGS)

After initializing $r = -H_0 \mathcal{F}(\mathbf{z}_0)$

BFGS Iteration

First update:
for i to $\min\{m, k\}$

- Calculate $\gamma_i = \rho_i s_i^\top r$
- Update residual $r = r - \gamma_i y_i$

Second update:
for i to $\min\{m, k\}$

- Update search direction $d_k = d_k + s_i(\gamma_i - \rho_i y_i^\top d_k)$

Update $\mathbf{z}_{k+1} = \mathbf{z}_k + a_k d_k$

$$s_k = \mathbf{z}^k - \mathbf{z}_{k-1} \qquad y_k = \mathcal{F}(\mathbf{z}_k) - \mathcal{F}(\mathbf{z}_{k-1})$$

$$\rho_k = \left(s_k y_k^\top\right)^{-1} \qquad H_{k+1} = (I - \rho_k y_k s_k^\top)^\top H_k (I - \rho_k y_k s_k^\top)$$

Preconditioned Limited Memory BFGS (P-L-BFGS)

After initializing $r = -H_0\mathcal{F}(\mathbf{z}_0)$

BFGS Iteration

First update:

for i to $\min\{m, k\}$

- Calculate $\gamma_i = \rho_i s_i^\top r$
- Update residual $r = r - \gamma_i y_i$

Second update:

for i to $\min\{m, k\}$

- Update search direction $d_k = d_k + s_i(\gamma_i - \rho_i y_i^\top d_k)$

Update $\mathbf{z}_{k+1} = \mathbf{z}_k + a_k d_k$

$$s_k = \mathbf{z}^k - \mathbf{z}_{k-1} \qquad y_k = \mathcal{F}(\mathbf{z}_k) - \mathcal{F}(\mathbf{z}_{k-1})$$

$$\rho_k = \left(s_k y_k^\top\right)^{-1} \qquad H_{k+1} = (I - \rho_k y_k s_k^\top)^\top H_k (I - \rho_k y_k s_k^\top)$$

For our tests we have choose to set $m = 7$, $H_0^{-1} = A$.

Numerical Results: 1D - Formulation

The Vornonoi regions are simple to construct

$$\mathcal{V}_i = (d_{i-1}, d_{i+1}) = (\frac{z_{i-1} + z_i}{2}, \frac{z_i + z_{i+1}}{2}).$$

The energy function given by the variance is defined by

$$E(z, d(z)) = \sum_{i=1}^{n} \int_{d_{i-1}}^{d_i} \|x - z_i\|^2 \rho(x)\ dx.$$

The gradient is

$$\partial_{z_i} E = 2 \int_{d_{i-1}}^{d_i} (z_i - x)\rho(x)\ dx.$$

Numerical Results: 1D - Formulation

The Hessian is

$$\frac{\partial^2 E}{\partial z_i \partial z_{i-1}} = -\frac{1}{2}\rho(d_{i-1})\,(z_i - z_{i-1})\,,$$

$$\frac{\partial^2 E}{\partial z_i \partial z_{i+1}} = -\frac{1}{2}\rho(d_i)\,(z_{i+1} - z_i)$$

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

Numerical Results: 1D - Formulation

The Hessian is

$$\frac{\partial^2 E}{\partial z_i \partial z_{i-1}} = -\frac{1}{2}\rho(d_{i-1})\left(z_i - z_{i-1}\right),$$

$$\frac{\partial^2 E}{\partial z_i \partial z_{i+1}} = -\frac{1}{2}\rho(d_i)\left(z_{i+1} - z_i\right)$$

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

The Graph Laplacian: (with suitable modification near the boundary generators)

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

1-D: Numerical Methods

For the initial guess we implement a two-grid method. Starting from a coarse, say 32, uniformly distributed generators.

- Lloyd relaxtion on the coarse grid until $\|32\mathcal{F}(z_k)\| < 1.$e-6.
- Refine to fine grids by consecutive midpoints.

1-D: Numerical Methods

For the initial guess we implement a two-grid method. Starting from a coarse, say 32, uniformly distributed generators.

- Lloyd relaxtion on the coarse grid until $\|32\mathcal{F}(z_k)\| < 1.e\text{-}6$.
- Refine to fine grids by consecutive midpoints.

Quasi-Newton method

$$\mathbf{z}^{k+1} = \mathbf{z}^k - A^{-1}\nabla E(\mathbf{z}^k).$$

1-D: Numerical Methods

For the initial guess we implement a two-grid method. Starting from a coarse, say 32, uniformly distributed generators.

- Lloyd relaxtion on the coarse grid until $\|32\mathcal{F}(z_k)\| < 1.\text{e-}6$.
- Refine to fine grids by consecutive midpoints.

Quasi-Newton method

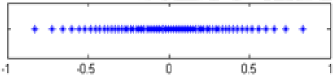
$$\mathbf{z}^{k+1} = \mathbf{z}^k - A^{-1}\nabla E(\mathbf{z}^k).$$

Number of Generators tested 2^L, L $=$ 8 to 20.
Stopping Criteria $\|2^L\mathcal{F}(z_k)\|_\infty < 1.\text{e-}12$

Numerical Results: 1D - Gaussian

Gaussian distribution

$\rho(x) = e^{-10x^2}$ on $\Omega = [-1, 1]$



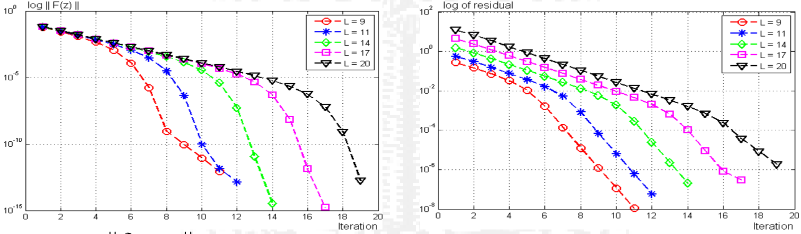
$\#$ of generators $= 2^L$

Error Toloreance $=$ 1.e-12

L	Iter	$\|2^L \mathcal{F}(z)\|_\infty$
8	12	3.8243e-013
9	12	3.0726e-013
10	12	2.7922e-013
11	12	2.7887e-013
12	13	3.2442e-014
13	14	4.5251e-015
14	15	1.6482e-015
15	16	1.6482e-015
16	17	1.6482e-015
17	17	6.7942e-013
18	18	3.3992e-013
19	19	1.6975e-013
20	20	8.4876e-014

Numerical Results: 1D - Gaussian

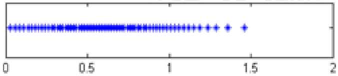
Gauss distribution: $\rho(x) = e^{-10x^2}$ on $\Omega = (-1, 1)$



RRF $= \frac{\|\delta z_{k+1}\|}{\|\delta z_k\|}$, # of generators $= 2^L$, Error Toloreance $=$ 1.e-12

Numerical Results: 1D - Weibull

Weibull distribution: $\rho(x) = 6x^2 e^{-2x^3}$ on $\Omega = (0, 2)$

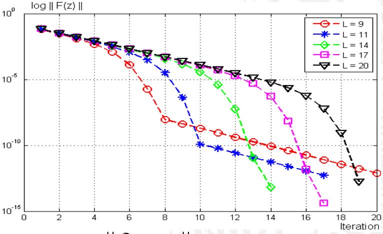


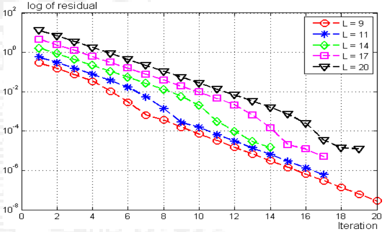
$\#$ of generators $= 2^L$
Error Toloreance = 1.e-12

L	Iter	$\|2^L \mathcal{F}(z)\|_\infty$
8	14	3.7231e-013
9	13	2.6792e-013
10	12	9.8253e-013
11	12	6.7196e-013
12	13	9.8956e-014
13	14	1.1654e-014
14	15	4.1452e-015
15	16	3.6845e-015
16	17	3.9958e-015
17	17	8.4150e-013
18	18	4.1955e-013
19	19	2.1177e-013
20	20	1.0389e-013

Numerical Results: 1D - Weibull

Weibull distribution: $\rho(x) = 6x^2 e^{-2x^3}$ on $\Omega = (0, 2)$

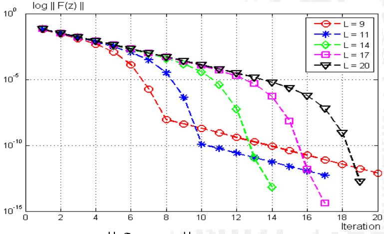


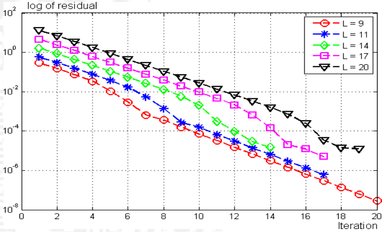


RRF $= \dfrac{\|\delta z_{k+1}\|}{\|\delta z_k\|}$, # of generators $= 2^L$, Error Toloreance $=$ 1.e-12.

Numerical Results: 1D - Weibull

Weibull distribution: $\rho(x) = 6x^2 e^{-2x^3}$ on $\Omega = (0, 2)$





RRF $= \dfrac{\|\delta z_{k+1}\|}{\|\delta z_k\|}$, # of generators $= 2^L$, Error Toloreance $=$ 1.e-12.

It is optimal but not as good as FAS. In Koren, Yaveneh and Spira 2005, the RRF is $0.17 - 0.20$.

Numerical Results: 2D - Outline of test

For each test we start with random distributed generators with respect to the given density function.

nFeval is the number of evalutations of the energy function.

$\mathcal{E}$ is the final energy.

$\|D^{-1}\nabla\mathcal{E}\|$ is the L_2 norm of the weighted error (problem size independent). D is the matrix of masses of the Voronoi tessellation.

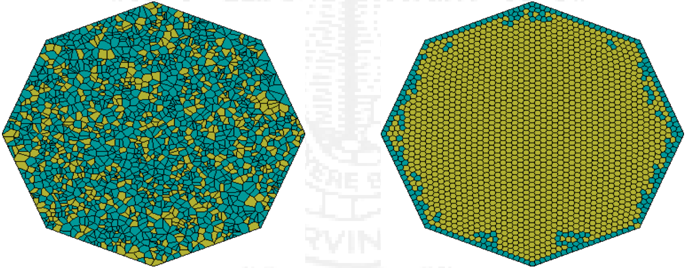
$\|\nabla\mathcal{E}\|$ is the L_2 norm of the error.

Stopping criteria is set at $\|D^{-1}\nabla\mathcal{E}\| < 1.e-6$.

2-D Constant Distribution

$\rho = 1$ with 2000 generators. $\Omega =$ A regular octagon bounded by $[-2, 2] \times [-2, 2]$

Method	iter.	nFeval	Time(seconds)	$\mathcal{E}$	$\|\|D^{-1}\nabla\mathcal{E}\|\|$	$\|\|\nabla\mathcal{E}\|\|$
Lloyd	1000	1000	34.17	1.037639e-02	1.792159e-03	9.789437e-06
L-BFGS(7)	408	438	15.61	1.036375e-02	9.383625e-07	7.066382e-08
P-L-BFGS(7)	285	308	15.10	1.036287e-02	8.329667e-07	6.236222e-08
NLCG	290	300	17.57	1.037063e-02	9.824495e-07	7.415228e-08
P-NLCG	203	224	17.85	1.036535e-02	9.686512e-07	7.269889e-08



Fast Computation for CVTs

Introduction

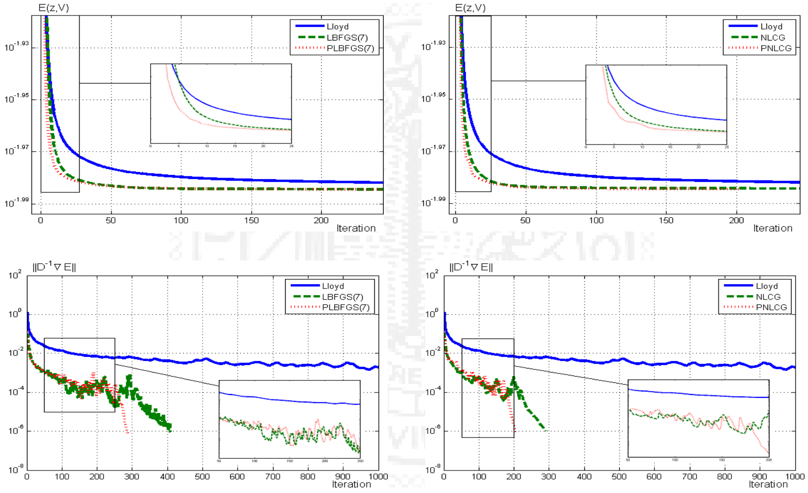
Preliminaries

Graph Laplacian

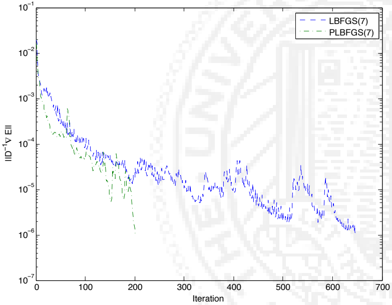
Optimization Methods

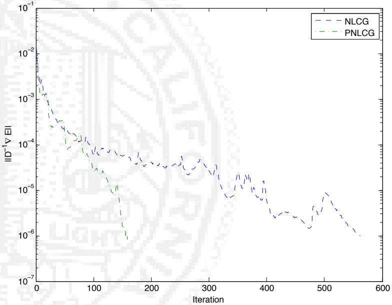
Numerical Results

Constant Distribution: Energy and Error



Error Comparison

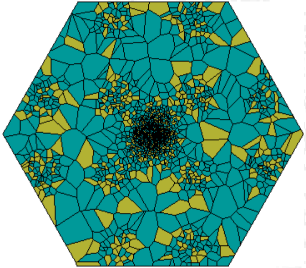


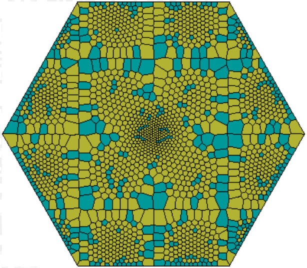


2D - Non-constant Distribution

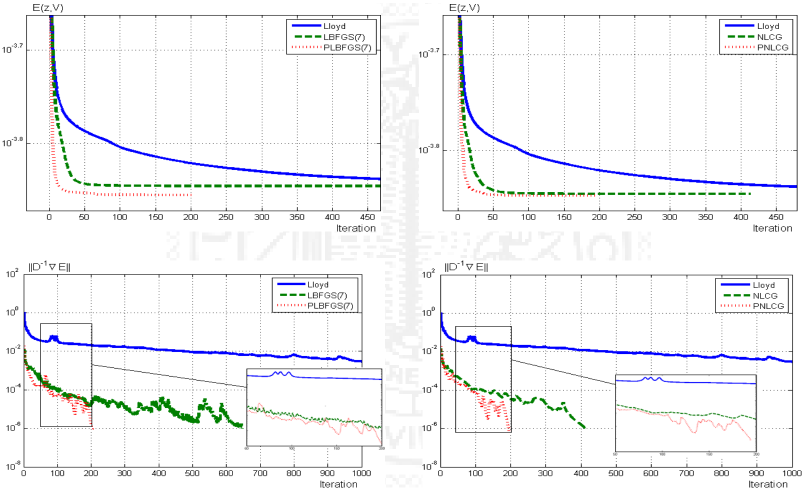
$\rho(x,y) = e^{-20(x^2+y^2)} + \frac{1}{20}\sin^2(\pi x)\sin^2(\pi y)$, 2000 generators.
Ω = a regular hexagon bounded by $[-2,2] \times [-1.732, 1.732]$.

Method	iter	nFeval	Time(seconds)	$\mathcal{E}$	$\|\|D^{-1}\nabla\mathcal{E}\|\|$	$\|\|\nabla\mathcal{E}\|\|$
Lloyd	1000	1000	55.12	1.435175e-04	2.955707e-03	2.923185e-07
LBFGS(7)	647	679	43.26	1.428238e-04	9.957305e-07	9.643630e-09
PLBFGS(7)	202	208	15.66	1.397377e-04	9.233494e-07	1.211738e-08
NLCG	413	413	38.26	1.427143e-04	9.864819e-07	1.145847e-08
PNLCG	194	207	24.02	1.421614e-04	7.533385e-07	1.125569e-08

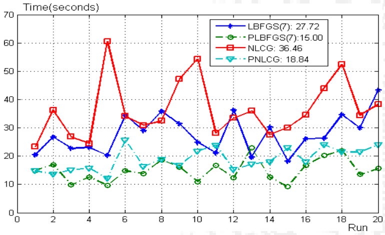


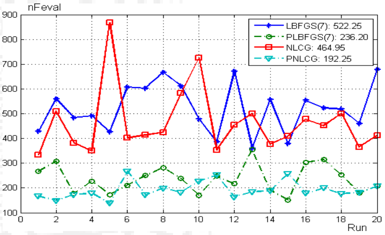


Energy and Error



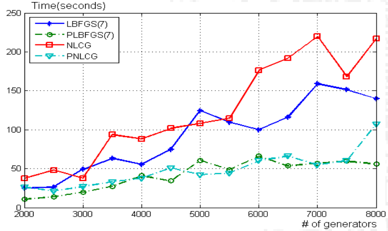
Time and Iteration Steps

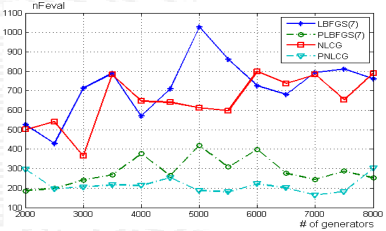




Independent of Problem Size

To test robustness we test 2000 to 8000 generators with an increment of 500. The stopping criteria is $\|D^{-1}\nabla\mathcal{E}\| <$1.e-6 (independent of problem size).

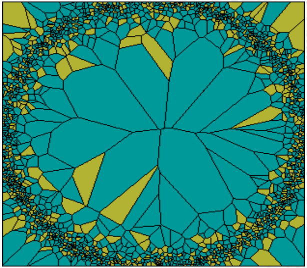


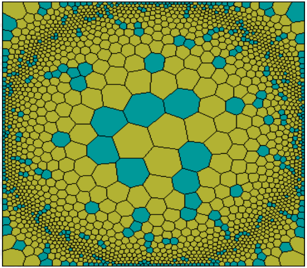


2D - Non-constant Distribution

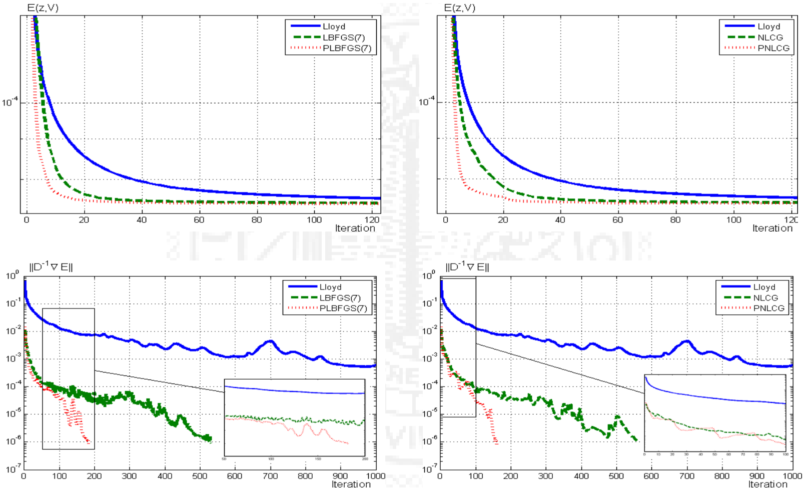
$\rho(x, y) = e^{-10|x^2+y^2-1|}$ with 2000 generators.
$\Omega = (-1, 1) \times (-1, 1)$.

Method	iter	nFeval	Time(seconds)	E	$\|\|D^{-1}\nabla E\|\|$	$\|\|\nabla E\|\|$
Lloyd	1000	1000	72.27	7.471462e-05	6.113908e-04	1.540093e-07
LBFGS(7)	530	547	39.60	7.455624e-05	9.091596e-07	6.983997e-09
PLBFGS(7)	182	220	19.00	7.457583e-05	8.881937e-07	1.515717e-08
NLCG	562	566	56.72	7.458000e-05	9.710043e-07	9.620996e-09
PNLCG	159	171	18.09	7.450278e-05	8.156436e-07	1.567448e-08

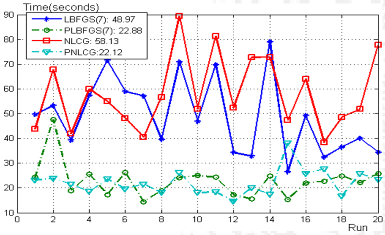


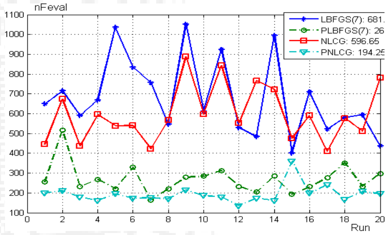


Energy and Error



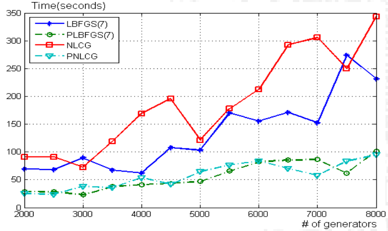
Time and Iteration Steps

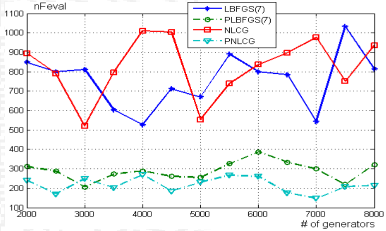




Independent of Problem Size

To test robustness we test 2000 to 8000 generators with an increment of 500. The stopping criteria is $\|D^{-1}\nabla\mathcal{E}\| <$1.e-6 (independent of problem size).





Two-Grid Method

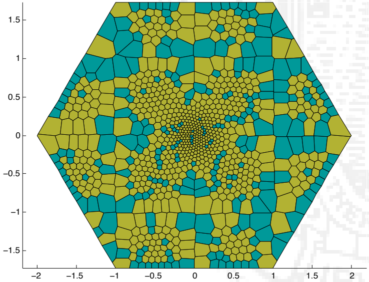
To get an good initial guess we implement a two-grid method.

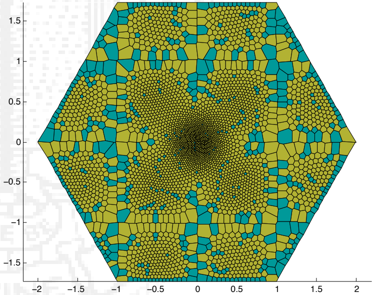
- P-L-BFGS on the coarse grid.
- Refine to fine grids by uniform refinement.
- P-L-BFGS on the fine grid.

Two-Grid Method

To get an good initial guess we implement a two-grid method.

- P-L-BFGS on the coarse grid.
- Refine to fine grids by uniform refinement.
- P-L-BFGS on the fine grid.

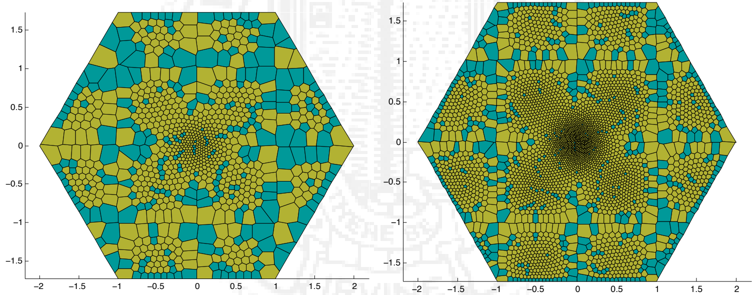




Two-Grid Method

To get an good initial guess we implement a two-grid method.

- P-L-BFGS on the coarse grid.
- Refine to fine grids by uniform refinement.
- P-L-BFGS on the fine grid.



Work in progress: Adaptive Multiscale Redistribution by Koren and Yavneh 2006.

Thank you for your attention!

Thank you for your attention!

Supported by NSF.