ConvexVST: A Convex Optimization Approach to Variance-stabilizing transformation

Appendix

Contents

1. Proof of Lemma 1

In the paper, we used new variables Δx_k to represent the distances between adjacent samples: $\Delta x_k = f(x_k) - f(x_{k-1})$ so that we directly optimize Δx instead of the transform function $f(x)$. Oppositely, every transformation is the sum of several variables that $f(x_k) = \sum_{j=1}^{k} \Delta x_j + x_0$. The signal θ can be processed in the same way when θ is also in the sample space. Otherwise, we can assume $x_{l-1} \leq \theta \leq x_l$ so that θ can be represented by a linear combination: $f(\theta) = \sum_{j=1}^{l-1} \Delta x_j + x_0 + \gamma_\theta \Delta x_l$ where $0 \leq \gamma_\theta \leq 1$ if $\theta \notin \mathcal{S}_x$.

The objective function can be represented with Δx_k:

$$
\begin{aligned}
C'_f &= \max_\theta \sigma^2_{\theta,f} && (1)\\
&= \max_\theta E\{[f(X_\theta) - f(\theta)]^2\} && (2)\\
&= \max_\theta \{\sum_{k=0}^{m-1} p_{\theta k}[f(x_k) - f(\theta)]^2\} && (3)\\
&= \max_\theta \{\sum_k p_{\theta k}[\sum_{j=1}^{k} \Delta x_j - \sum_{j=1}^{l-1} \Delta x_j - \gamma_\theta \Delta x_l]^2\} && (4)
\end{aligned}
$$

$\sigma^2_{\theta,f}$ is the combination of a set of quadratic functions with positive coefficients $p_{\theta k}$, which is always convex.

Then we show that the maximum of a set of convex functions is also convex. Because $\sigma^2_{\theta,f}$ is convex, we have $\sigma^2_{\theta,f}[tx + (1-t)y] \leq t\sigma^2_{\theta,f}(x) + (1-t)\sigma^2_{\theta,f}(y)$ for all $x, y \in \mathbf{dom} f, 0 \leq t \leq 1$. We assume θ_n is a choice of Θ. For θ_n:

$$
\begin{aligned}
\sigma^2_{\theta_n,f}[tx + (1-t)y] &\leq t\sigma^2_{\theta_n,f}(x) + (1-t)\sigma^2_{\theta_n,f}(y) && (5)\\
&\leq \max_\theta \{t\sigma^2_{\theta,f}(x) + (1-t)\sigma^2_{\theta,f}(y)\} && (6)\\
&\leq t \max_\theta \{\sigma^2_{\theta,f}(x)\} + (1-t) \max_\theta \{\sigma^2_{\theta,f}(y)\} && (7)
\end{aligned}
$$

So is it for all other θ. Thus,

$$
\max_\theta \{\sigma^2_{\theta,f}[tx + (1-t)y]\} \leq t \max_\theta \{\sigma^2_{\theta,f}(x)\} + (1-t) \max_\theta \{\sigma^2_{\theta,f}(y)\} \quad (8)
$$

As a result, $\max_\theta \sigma^2_{\theta,f}$ is also convex.

2. Proof of Lemma 2

We can represent the $\sigma^2_{\theta,f}$ in the matrix form:

$$
\begin{aligned}
\sigma^2_{\theta,f} &= \sum_{k=0}^{m-1} p_{\theta k}[\sum_{j=1}^{k} \Delta x_j - \sum_{j=1}^{l-1} \Delta x_j - \gamma_\theta \Delta x_l]^2 && (9)\\
&= \Delta \mathbf{x}^T \mathbf{Q}_\theta \Delta \mathbf{x} && (10)
\end{aligned}
$$

Form 9 can be converted into Form 10 by:

$$
\mathbf{Q}_\theta\{ij\} = \begin{cases}
\sum_{k=0}^{\min\{i,j\}} p_{\theta k} & \max\{i,j\} < l \\
\sum_{k=\max\{i,j\}}^{m-1} p_{\theta k} & \min\{i,j\} > l \\
\gamma_\theta \sum_{k=0}^{\min\{i,j\}} p_{\theta k} & \max\{i,j\} = l \wedge i \neq j \\
(1-\gamma_\theta) \sum_{k=\max\{i,j\}}^{m-1} p_{\theta k} & \min\{i,j\} = l \wedge i \neq j \\
r_\theta^2 \sum_{k=0}^{l-1} p_{\theta k} + (1-\gamma_\theta)^2 \sum_{l}^{m-1} p_{\theta k} & i = j = l,
\end{cases} \quad (11)
$$

where $\mathbf{Q}_\theta\{ij\}$ is the entry at the i_{th} row and j_{th} column of $\mathbf{Q}_\theta$. $\mathbf{Q}_\theta$ is a positive definite matrix because $\sigma^2_{\theta,f}$ is quadratic and always positive.

As a positive definite matrix, $\mathbf{Q}_\theta$ can be decomposed to the product of a matrix and its transpose $\mathbf{Q}_\theta = \mathbf{F}_\theta^T \mathbf{F}_\theta$ by methods like Cholesky decomposition. Actually, we can derive $\mathbf{F}_\theta$ directly without computation:

$$\mathbf{F}_\theta = \begin{bmatrix} \sqrt{p_{\theta 0}} & \sqrt{p_{\theta 0}} & \sqrt{p_{\theta 0}} & \dots & \sqrt{p_{\theta 0}} & \gamma_\theta\sqrt{p_{\theta 0}} & \dots & 0 & 0 & 0 \\ 0 & \sqrt{p_{\theta 1}} & \sqrt{p_{\theta 1}} & \dots & \sqrt{p_{\theta 1}} & \gamma_\theta\sqrt{p_{\theta 1}} & \dots & 0 & 0 & 0 \\ 0 & 0 & \sqrt{p_{\theta 2}} & \dots & \sqrt{p_{\theta 2}} & \gamma_\theta\sqrt{p_{\theta 2}} & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \sqrt{p_{\theta(l-1)}} & \gamma_\theta\sqrt{p_{\theta(l-1)}} & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & (1-\gamma_\theta)\sqrt{p_{\theta l}} & \dots & 0 & 0 & 0 \\ \vdots & \vdots & & & \vdots & \vdots & & \vdots & \vdots & \vdots \\ \vdots & \vdots & 0 & \dots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & (1-\gamma_\theta)\sqrt{p_{\theta(m-3)}} & \dots & \sqrt{p_{\theta(m-3)}} & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & (1-\gamma_\theta)\sqrt{p_{\theta(m-2)}} & \dots & \sqrt{p_{\theta(m-2)}} & \sqrt{p_{\theta(m-2)}} & 0 \\ 0 & 0 & 0 & \dots & 0 & (1-\gamma_\theta)\sqrt{p_{\theta(m-1)}} & \dots & \sqrt{p_{\theta(m-1)}} & \sqrt{p_{\theta(m-1)}} & \sqrt{p_{\theta(m-1)}} \end{bmatrix}, \tag{12}$$

where

$$\mathbf{F}_\theta\{ij\} = \begin{cases} \sqrt{p_{\theta i}} & (i < l \wedge j \le i) | (i > l \wedge j \ge i) \\ \gamma_\theta\sqrt{p_{\theta_i}} & i = l \wedge j < l \\ (1-\gamma_\theta)\sqrt{p_{\theta_i}} & i = l \wedge j \ge l \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

The inverse of $\mathbf{F}_\theta$ can be directly derived as well:

$$\mathbf{F}_\theta^{-1} = \begin{bmatrix} \frac{1}{\sqrt{p_{\theta 0}}} & -\frac{1}{\sqrt{p_{\theta 1}}} & 0 & \dots & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{p_{\theta 1}}} & -\frac{1}{\sqrt{p_{\theta 2}}} & \dots & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{p_{\theta 2}}} & \dots & 0 & 0 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \frac{1}{\sqrt{p_{\theta(l-1)}}} & -\frac{\gamma_\theta}{(1-\gamma_\theta)\sqrt{p_{\theta(l-1)}}} & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & \frac{1}{(1-\gamma_\theta)\sqrt{p_{\theta l}}} & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & -\frac{1}{\sqrt{p_{\theta l}}} & \dots & 0 & 0 & 0 \\ \vdots & \vdots & & & \vdots & \vdots & & \vdots & \vdots & \vdots \\ \vdots & \vdots & 0 & \dots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 0 & \dots & -\frac{1}{\sqrt{p_{\theta(m-3)}}} & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & \dots & 0 & \frac{1}{\sqrt{p_{\theta(m-2)}}} & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & \dots & 0 & -\frac{1}{\sqrt{p_{\theta(m-2)}}} & \frac{1}{\sqrt{p_{\theta(m-1)}}} \end{bmatrix}, \tag{14}$$

where

$$\mathbf{F}_\theta\{ij\} = \begin{cases} \frac{1}{\sqrt{p_{\theta i}}} & i = j \neq l \\ -\frac{1}{\sqrt{p_{\theta j}}} & (i-1 = j < l) | (i+1 = j \ge l) \\ -\frac{\gamma_\theta}{(1-\gamma_\theta)\sqrt{p_{\theta(l-1)}}} & i-1 = j = l \\ \frac{1}{(1-\gamma_\theta)\sqrt{p_{\theta l}}} & i = j = l \\ 0 & otherwise \end{cases} \tag{15}$$

3. Proof of Lemma 3

In this part, we will give a brief introduction about the second-order cone programming (SOCP) problem and show how our problems can be solved efficiently.

Definition 1 – second-order cone. A $(n+1)$-dimensional second-order cone is defined as

$$\mathcal{Q}^{n+1} = \{(\mathbf{x}, r) \in \mathbb{R}^{n+1} | \, ||\mathbf{x}||_2 \le r\} \tag{16}$$

Definition 2 – second-order conic constraint. A second-order conic constraint has the form

$$||\mathbf{A}\mathbf{x} + \mathbf{b}||_2 \le \mathbf{c}^T\mathbf{x} + d, \tag{17}$$

which is equivalent to requiring the affine function $(||\mathbf{A}\mathbf{x} + \mathbf{b}||_2, \mathbf{c}^T\mathbf{x} + d)$ to lie in the second-order cone in $\mathcal{Q}^{n+1}$. $\mathbf{x}$ is the variable and $\mathbf{A}, \mathbf{b}, \mathbf{c}, d$ are constants.

Definition 3 – SOCP problem. A SOCP problem is a convex optimization problem of the form:

$$\min \quad \mathbf{f}^T\mathbf{x} \tag{18}$$
$$s.j. \quad ||\mathbf{A}_i\mathbf{x}+\mathbf{b}_i||_2 \le \mathbf{c}_i^T\mathbf{x}+d \tag{19}$$
$$\mathbf{F}\mathbf{x}=\mathbf{g}, \tag{20}$$

which minimizes a linear function subjected to second-order conic constraints and linear constraints.

Here we only illustrate the first part of the algorithm, while the second part of the algorithm can be reformulated in the same way:

$$\min\max_{\theta} \quad \Delta\mathbf{x}^T\mathbf{Q}_\theta\Delta\mathbf{x} \tag{21}$$
$$s.j. \quad \Delta\mathbf{x}\ge\mathbf{0} \tag{22}$$
$$\Delta\mathbf{x}^T\mathbf{1}=x_{m-1}-x_0 \tag{23}$$

It can be rewritten in the form:

$$\min \quad t^2 \tag{24}$$
$$s.j. \quad \Delta\mathbf{x}^T\mathbf{Q}_\theta\Delta\mathbf{x}\le t^2 \tag{25}$$
$$\Delta\mathbf{x}\ge\mathbf{0} \tag{26}$$
$$\Delta\mathbf{x}^T\mathbf{1}=x_{m-1}-x_0 \tag{27}$$

The two forms are equal, have the same feasible region, and the same optimal solution. However, $\Delta\mathbf{x}^T\mathbf{Q}_\theta\Delta\mathbf{x}-t^2$ is non-convex and the second form isn't a convex optimization problem. This kind of problem is called the "abstract convex optimization problem" (Boyd & Vandenberghe, 2004): even it doesn't follow the definition of the convex optimization problem, the feasible region is also convex. It can be converted into a convex optimization problem and solved by convex optimization solvers.

In Section 2 we showed that $\mathbf{Q}_\theta$ had a special decompostion. After the decomposition, we can convert $\Delta\mathbf{x}^T\mathbf{Q}_\theta\Delta\mathbf{x}=\Delta\mathbf{x}^T\mathbf{F}_\theta^T\mathbf{F}_\theta\Delta\mathbf{x}=||\mathbf{F}_\theta\Delta\mathbf{x}||_2^2$, and $\Delta\mathbf{x}^T\mathbf{Q}_\theta\Delta\mathbf{x}\le t^2$ is equivalent to $||\mathbf{F}_\theta\Delta\mathbf{x}||_2\le t$. So the optimization problem equals to

$$\min \quad t \tag{28}$$
$$s.j. \quad ||\mathbf{F}_\theta\Delta\mathbf{x}||_2\le t \tag{29}$$
$$\Delta\mathbf{x}\ge\mathbf{0} \tag{30}$$
$$\Delta\mathbf{x}^T\mathbf{1}=x_{m-1}-x_0, \tag{31}$$

which is a SOCP problem.

Definition 4 – dual cone. For a subspace $K\subset\mathbb{R}^n$, $K_*:=\{\mathbf{s}:\mathbf{x}\in K|\mathbf{s}^T\mathbf{x}\ge 0\}$ is the dual cone of K.

K is self-dual if K is a closed convex cone and $K=K_*$.

Definition 5- homogeneous cone. A closed convex cone $K\subset\mathbb{R}^n$ is homogeneous if for any $\mathbf{x},\mathbf{s}\in int(K)$ there exists $\mathbf{B}\in\mathbb{R}^{n\times n}$ so that $\mathbf{B}(K)=K$and $\mathbf{B}\mathbf{x}=\mathbf{s}$, where $int(K)$ denotes the interior of K.

Lemma 1. *Two kinds of cones, $\mathbb{R}_+:=\{x\in\mathbb{R}|x\ge 0\}$ and the second-order cone are self-dual and homogeneous (Andersen et al., 2003).*

Lemma 2. *If K_1 and K_2 is homogeneous self-dual, the direct product of the two cones, $K:=K_1\times K_2$ is still homogeneous self-dual (Nesterov & Todd, 1997).*

Every second-order constraint $||\mathbf{F}_\theta\Delta\mathbf{x}||_2\le t$ is equivalent to a linear equality constraint $\mathbf{F}_\theta\Delta\mathbf{x}=\mathbf{g}_\theta$ requiring the affine function $(\mathbf{g}_\theta,t)$ lying in the second-order cone. The constraint $\Delta\mathbf{x}\ge\mathbf{0}$ is equivalent to constrain every variable in $\mathbb{R}_+$. Then the SOCP problem can be solved efficiently by (Andersen et al., 2003) when converted to the conic quadratic optimization

problem:

$$\begin{aligned} \min \quad & t && (32)\\ s.j. \quad & \mathbf{F}_\theta \Delta\mathbf{x} = \mathbf{g}_\theta && (33)\\ & \Delta\mathbf{x}^T\mathbf{1} = x_{m-1} - x_0 && (34)\\ & (\mathbf{g}_\theta, t) \in \mathcal{Q}^{m+1} && (35)\\ & \Delta\mathbf{x} \in \mathbb{R}_+^m, && (36) \end{aligned}$$

which has linear equality constraints and homogeneous self-dual cones only.

Supposing the size of Θ and $\mathcal{S}_x$ is n and m, the problem can be solved in $\mathcal{O}(nm)$ time complexity operations except finding the inverse of $\mathbf{F^T}(\mathbf{\Theta W})^{-2}\mathbf{F} \in \mathbb{R}^{nm\times nm}$, where $\mathbf{F} = diag([\mathbf{F}_0 \quad \mathbf{F}_1 \dots \mathbf{F}_{m-1}])$ and $\mathbf{\Theta}$ and $\mathbf{W}$ are two matrices already known. The key step is the bottleneck of off-the-shelf convex solvers, which limits the time complexity to $\mathcal{O}(n^3m^3)$. However, as mentioned in Section 2, the inverse of $\mathbf{F}$ can be derived directly, while it's multiplication to any matrix can be processed in $\mathcal{O}(nm)$ steps because of its sparsity so that it's total time complexity can be decreased to $\mathcal{O}(nm)$.

4. Proof of Lemma 4

We restate the second part of the algorithm at the beginning. Firstly, we define H

$$\begin{aligned} H(\Delta\mathbf{x}) &= \max_\theta |\sigma^2_{\theta,f} - c| && (37)\\ &= \max_\theta \{\sigma^2_{\theta,f} - c, c - \sigma^2_{\theta,f}\} && (38)\\ &= \max_\theta \{\Delta\mathbf{x}^T\mathbf{Q}_\theta\Delta\mathbf{x} - c, c - \Delta\mathbf{x}^T\mathbf{Q}_\theta\Delta\mathbf{x}\}, && (39) \end{aligned}$$

is the original objective function, and h_k

$$\begin{aligned} h_k(\Delta\mathbf{x}) &= \max_\theta \{\sigma^2_{\theta,f} - c, c - \sigma^2_{\theta,f}(\Delta\mathbf{x}_k) - \frac{\partial\sigma^2_{\theta,f}}{\partial\Delta\mathbf{x}}(\Delta\mathbf{x}_k)(\Delta\mathbf{x} - \Delta\mathbf{x}_k)\} && (40)\\ &= \max_\theta \{\Delta\mathbf{x}^T\mathbf{Q}_\theta\Delta\mathbf{x} - c, c - \Delta\mathbf{x}_k^T\mathbf{Q}_\theta\Delta\mathbf{x}_k - \frac{\partial\Delta\mathbf{x}^T\mathbf{Q}_\theta\Delta\mathbf{x}}{\partial\Delta\mathbf{x}}(\Delta\mathbf{x}_k)(\Delta\mathbf{x} - \Delta\mathbf{x}_k)\} && (41)\\ &= \max_\theta \{\Delta\mathbf{x}^T\mathbf{Q}_\theta\Delta\mathbf{x} - c, -2\Delta\mathbf{x}_k^T\mathbf{Q}_\theta\Delta\mathbf{x} + \Delta\mathbf{x}_k^T\mathbf{Q}_\theta\Delta\mathbf{x}_k + c\}, && (42) \end{aligned}$$

is an approximation of $H(\Delta\mathbf{x})$ by substituting $c - \sigma^2_{\theta,f}$ with its first-order Taylor expansion at $\Delta\mathbf{x}_k$.

In the second part, we want to minimize the objective function H and argue that:

4.1. Convexity

Lemma 3. *h_k is always convex for any $\Delta\mathbf{x}_k$.*

In Section 1 we showed that $\sigma^2_{\theta,f} - c$ is convex. Because $c - \sigma^2_{\theta,f}(\Delta\mathbf{x}_k) - \frac{\partial\sigma^2_{\theta,f}}{\partial\Delta\mathbf{x}}(\Delta\mathbf{x}_k)(\Delta\mathbf{x} - \Delta\mathbf{x}_k)$ is the first-order Taylor expansion of a function, it's also convex. As a result, h_k is the maximum of a set of convex functions, whose convexity has been proved.

4.2. Convergence

Lemma 4. *If we solve the optimization problems iteratively, which minimize h_k under certain constraints, the result $h_k(\Delta\mathbf{x}_k)$ will converge.* In detail, with the initialization of one iteration $\Delta\mathbf{x}_k$, we have the optimization problem:

$$\begin{aligned} \min \quad & h_k && (43)\\ s.j. \quad & \Delta\mathbf{x} \geq \mathbf{0} && (44)\\ & \Delta\mathbf{x}^T\mathbf{1} = x_{m-1} - x_0 && (45) \end{aligned}$$

The optimal solution $\Delta\mathbf{x}_{k+1}$ is also the initialization of the next iteration. Then we will show $h_k(\Delta\mathbf{x}_k) \geq h_{k+1}(\Delta\mathbf{x}_{k+1})$ with any initialization of the first iteration $\Delta\mathbf{x}_0$ and the results can converge.

Because $c - \sigma^2_{\theta,f}$ is concave, we have

$$c - \sigma^2_{\theta,f} \leq c - \sigma^2_{\theta,f}(\Delta\mathbf{x}_k) - \frac{\partial\sigma^2_{\theta,f}}{\partial\Delta\mathbf{x}}(\Delta\mathbf{x}_k)(\Delta\mathbf{x} - \Delta\mathbf{x}_k) \tag{46}$$

$$c - \Delta\mathbf{x}^T\mathbf{Q}_\theta\Delta\mathbf{x} \leq -2\Delta\mathbf{x}_k^T\mathbf{Q}_\theta\Delta\mathbf{x} + \Delta\mathbf{x}_k^T\mathbf{Q}_\theta\Delta\mathbf{x}_k + c \tag{47}$$

for any $\Delta\mathbf{x}_k$ and all θ based on the definition of the concave function. Then we have:

$$\max_\theta\{c - \Delta\mathbf{x}^T\mathbf{Q}_\theta\Delta\mathbf{x}\} \leq \max_\theta\{-2\Delta\mathbf{x}_k^T\mathbf{Q}_\theta\Delta\mathbf{x} + \Delta\mathbf{x}_k^T\mathbf{Q}_\theta\Delta\mathbf{x}_k + c\} \tag{48}$$

$$\max_\theta\{\Delta\mathbf{x}^T\mathbf{Q}_\theta\Delta\mathbf{x} - c, c - \Delta\mathbf{x}^T\mathbf{Q}_\theta\Delta\mathbf{x}\} \leq \max_\theta\{\Delta\mathbf{x}^T\mathbf{Q}_\theta\Delta\mathbf{x} - c, -2\Delta\mathbf{x}_k^T\mathbf{Q}_\theta\Delta\mathbf{x} + \Delta\mathbf{x}_k^T\mathbf{Q}_\theta\Delta\mathbf{x}_k + c\} \tag{49}$$

$$H(\Delta\mathbf{x}) \leq h_k(\Delta\mathbf{x}) \tag{50}$$

It shows that h_k is always larger or equal to H in any cases. So we have $h_k(\Delta\mathbf{x}_{k+1}) \geq H(\Delta\mathbf{x}_{k+1})$. We also know that h_{k+1} is the first-order Taylor expansion of H at $\Delta\mathbf{x}_{k+1}$. Therefore $h_{k+1}(\Delta\mathbf{x}_{k+1}) = H(\Delta\mathbf{x}_{k+1})$.

Besides, $\Delta\mathbf{x}_k$ is the initialization of the problem minimizing h_k, and $\Delta\mathbf{x}_{k+1}$ is the optimal solution. Certainly, $h_k(\Delta\mathbf{x}_k) \geq h_k(\Delta\mathbf{x}_{k+1})$. In conclusion, we have:

$$h_k(\Delta\mathbf{x}_k) \geq h_k(\Delta\mathbf{x}_{k+1}) \geq H(\Delta\mathbf{x}_{k+1}) = h_{k+1}(\Delta\mathbf{x}_{k+1}) \tag{51}$$

So $\{h_k(\Delta\mathbf{x}_k)\}$ is a monotonic decreasing sequence of numbers and has a lower bound ($h_k \geq H \geq 0$). It can converge to its infimum based on the monotone convergence theorem.

4.3. Converge to the Local Optimum

Lemma 5. *This series of iterative problems are equivalent to the original optimization problem:*

$$\min \quad H \tag{52}$$

$$s.j. \quad \Delta\mathbf{x} \geq \mathbf{0} \tag{53}$$

$$\Delta\mathbf{x}^T\mathbf{1} = x_{m-1} - x_0 \tag{54}$$

To prove that the iterative problems equal to the original problem, we only need to show that two problems have exactly the same terminal condition. More narrowly, it will always stay at a local optimal solution of the original problem when the iteration terminates, and any local optimal solution can be the condition to terminate the iteration.

Firstly, we will show that any local optimal solution can be the condition to terminate the iteration. If a solution $\Delta\mathbf{x}^*$ is a local optimal solution of the original problem, there exists a real number $\epsilon > 0$ such that we have $H(\Delta\mathbf{x}^* + \boldsymbol{\delta}) > H(\Delta\mathbf{x}^*)$ for any $\boldsymbol{\delta}$ satisfying $||\boldsymbol{\delta}||_2 < \epsilon$. Assume $\Delta\mathbf{x}^*$ is the initialization of one iteration with the objective function h_k, we have $h_k \geq H$ and $h_k(\Delta\mathbf{x}^*) = H(\Delta\mathbf{x}^*)$ proved in 4.2. So $h_k(\Delta\mathbf{x}^* + \boldsymbol{\delta}) \geq H(\Delta\mathbf{x}^* + \boldsymbol{\delta}) > H(\Delta\mathbf{x}^*) = h_k(\Delta\mathbf{x}^*)$ for any $\boldsymbol{\delta}$ satisfying $||\boldsymbol{\delta}||_2 < \epsilon$. It means $\Delta\mathbf{x}^*$ is also the global optimal solution of minimizing h_k, and the iteration will terminate.

Then we show that the iterative calculation will always converge to a local optimal solution of the original problem. If the calculation converges when $\Delta\mathbf{x}^*$ is the initialization of one iteration with the objective function h_k, $\Delta\mathbf{x}^*$ is the global minimal solution and there exists a real number $\epsilon > 0$ such that we have $h_k(\Delta\mathbf{x}^* + \boldsymbol{\delta}) > h_k(\Delta\mathbf{x}^*)$ for any $\boldsymbol{\delta}$ satisfying $||\boldsymbol{\delta}||_2 < \epsilon$. We have:

$$h_k(\Delta\mathbf{x}) = \max_\theta\{\sigma^2_{\theta,f} - c, c - \sigma^2_{\theta,f}(\Delta\mathbf{x}^*) - \frac{\partial\sigma^2_{\theta,f}}{\partial\Delta\mathbf{x}}(\Delta\mathbf{x}^*)(\Delta\mathbf{x} - \Delta\mathbf{x}^*)\} \tag{55}$$

$$= \max_\theta\{\Delta\mathbf{x}^T\mathbf{Q}_\theta\Delta\mathbf{x} - c, -2\Delta\mathbf{x}^{*T}\mathbf{Q}_\theta\Delta\mathbf{x} + \Delta\mathbf{x}^{*T}\mathbf{Q}_\theta\Delta\mathbf{x}^* + c\}, \tag{56}$$

There are two cases. The first one is

$$h_k(\Delta\mathbf{x}^* + \boldsymbol{\delta}) = (\Delta\mathbf{x}^* + \boldsymbol{\delta})^T\mathbf{Q}_{\theta_n}(\Delta\mathbf{x}^* + \boldsymbol{\delta}) - c$$

for some certain θ_n. For this case, we have $h_k(\Delta\mathbf{x}^* + \boldsymbol{\delta}) = H(\Delta\mathbf{x}^* + \boldsymbol{\delta})$ because H also contains the same term and $H \leq h_k$. Then we have $H(\Delta\mathbf{x}^* + \boldsymbol{\delta}) = h_k(\Delta\mathbf{x}^* + \boldsymbol{\delta}) > h_k(\Delta\mathbf{x}^*) = H(\Delta\mathbf{x}^*)$. The second one is

$$h_k(\Delta\mathbf{x}^* + \boldsymbol{\delta}) = -2\Delta\mathbf{x}^{*T}\mathbf{Q}_{\theta_n}(\Delta\mathbf{x}^* + \boldsymbol{\delta}) + \Delta\mathbf{x}^{*T}\mathbf{Q}_{\theta_n}\Delta\mathbf{x}^* + c$$

for some certain θ_n. We have

$$h_k(\Delta\mathbf{x}^* + \boldsymbol{\delta}) - h_k(\Delta\mathbf{x}^*) = -2\Delta\mathbf{x}^{*T}\mathbf{Q}_{\theta_n}\boldsymbol{\delta} > 0. \tag{57}$$

And

$$H(\Delta\mathbf{x}^* + \boldsymbol{\delta}) - H(\Delta\mathbf{x}^*) = -2\Delta\mathbf{x}^{*T}\mathbf{Q}_{\theta_n}\boldsymbol{\delta} - \boldsymbol{\delta}^T\mathbf{Q}_{\theta_n}\boldsymbol{\delta} \tag{58}$$

is larger than zero when $||\boldsymbol{\delta}||_2$ is sufficient small because $-\boldsymbol{\delta}^T\mathbf{Q}_{\theta_n}\boldsymbol{\delta}$ is a higher-order infinitesimal compared to $-2\Delta\mathbf{x}^{*T}\mathbf{Q}_{\theta_n}\boldsymbol{\delta}$. So we can always find a small enough $\epsilon > ||\boldsymbol{\delta}||_2 > 0$ to make sure $H(\Delta\mathbf{x}^* + \boldsymbol{\delta}) > H(\Delta\mathbf{x}^*)$ when $h_k(\Delta\mathbf{x}^* + \boldsymbol{\delta}) > h_k(\Delta\mathbf{x}^*)$. It shows the iterative calculation always converges to a local optimal solution.

Besides, a local maximum of H can also terminate the iteration, but it's not stable and unlikely to happen for real application. This problem doesn't have saddle points.

In conclusion, the iterative calculation is to solve a series of convex optimization problems iteratively, which can always converge to a local optimal solution of the original problem.

5. Variance based Approach

In the paper, we shown the algorithm based on the MSE. Here we derive the algorithm based on the variance. We have:

$$\begin{aligned}
&Var[f(X_\theta)] && (59)\\
=&E\{[f(X_\theta) - E[f(X_\theta)]]^2\} && (60)\\
=&E[f^2(X_\theta)] - E^2[f(X_\theta)] && (61)\\
=&\sum_{k=0}^{m-1} p_{\theta k} f^2(x_k) - [\sum_{k=0}^{m-1} p_{\theta k} f(x_k)]^2 && (62)\\
=&\sum_{k=0}^{m-1} p_{\theta k}(x_0 + \sum_{j=1}^{k}\Delta x_j)^2 - [\sum_{k=0}^{m-1} p_{\theta k}(x_0 + \sum_{j=1}^{k}\Delta x_j)]^2 && (63)\\
=&\sum_{k=0}^{m-1} p_{\theta k}[x_0^2 + 2x_0\sum_{j=1}^{k}\Delta x_j + (\sum_{j=1}^{k}\Delta x_j)^2] - [(\sum_{k=0}^{m-1} p_{\theta k}x_0)^2 + 2(\sum_{k=0}^{m-1} p_{\theta k}x_0)(\sum_{k=1}^{m-1} p_{\theta k}\sum_{j=1}^{k}\Delta x_j) + (\sum_{k=1}^{m-1} p_{\theta k}\sum_{j=1}^{k}\Delta x_j)^2] && (64)\\
=&[x_0^2 + 2x_0\sum_{k=1}^{m-1} p_{\theta k}\sum_{j=1}^{k}\Delta x_j + \sum_{k=1}^{m-1} p_{\theta k}(\sum_{j=1}^{k}\Delta x_j)^2] - [x_0^2 + 2x_0\sum_{k=1}^{m-1} p_{\theta k}\sum_{j=1}^{k}\Delta x_j + (\sum_{k=1}^{m-1} p_{\theta k}\sum_{j=1}^{k}\Delta x_j)^2] && (65)\\
=&\sum_{k=1}^{m-1} p_{\theta k}(\sum_{j=1}^{k}\Delta x_j)^2 - (\sum_{k=1}^{m-1} p_{\theta k}\sum_{j=1}^{k}\Delta x_j)^2 && (66)\\
=&\Delta\mathbf{x}^T\mathbf{Q}_\theta\Delta\mathbf{x} && (67)
\end{aligned}$$

It shows the variance is also a quadratic function of $\Delta\mathbf{x}$ and can be converted into Form 67 by:

$$\begin{aligned}
Q_\theta\{ij\} &= \sum_{k=\max\{i,j\}}^{m-1} p_{\theta k} - (\sum_{k=i}^{m-1} p_{\theta k})(\sum_{k=j}^{m-1} p_{\theta k}) && (68)\\
&= \sum_{k=\max\{i,j\}}^{m-1} p_{\theta k}(1 - \sum_{k=\min\{i,j\}}^{m-1} p_{\theta k}) && (69)\\
&= \sum_{k=\max\{i,j\}}^{m-1} p_{\theta k}\sum_{k=0}^{\min\{i,j\}-1} p_{\theta k}, && (70)
\end{aligned}$$

where $\mathbf{Q}_\theta\{ij\}$ is the entry at the i_{th} row and j_{th} column of $\mathbf{Q}_\theta$. $\mathbf{Q}_\theta$ is positive definite because $Var[f(X_\theta)] > 0$. Then we formulate the problems based on different metrics in the same form so that they can be solved with the same algorithm.

In this case, it's hard to calculate the closed-form expression of a decomposition of $\mathbf{Q}_\theta$. Fortunately, we have another way to solve it efficiently. The inverse of $\mathbf{Q}_\theta$ has the closed-form:

$$\mathbf{Q}_\theta^{-1} = \begin{bmatrix} \frac{p_{\theta 0}+p_{\theta 1}}{p_{\theta 0}p_{\theta 1}} & -\frac{1}{p_{\theta 1}} & 0 & 0 & \dots & 0 & 0 \\ -\frac{1}{p_{\theta 1}} & \frac{p_{\theta 1}+p_{\theta 2}}{p_{\theta 1}p_{\theta 2}} & -\frac{1}{p_{\theta 2}} & 0 & \dots & 0 & 0 \\ 0 & -\frac{1}{p_{\theta 2}} & \frac{p_{\theta 2}+p_{\theta 3}}{p_{\theta 2}p_{\theta 3}} & -\frac{1}{p_{\theta 3}} & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & \frac{p_{\theta(m-3)}+p_{\theta(m-2)}}{p_{\theta(m-3)}p_{\theta(m-2)}} & -\frac{1}{p_{\theta(m-2)}} \\ 0 & 0 & 0 & 0 & \dots & -\frac{1}{p_{\theta(m-2)}} & \frac{p_{\theta(m-2)}+p_{\theta(m-1)}}{p_{\theta(m-2)}p_{\theta(m-1)}} \end{bmatrix}, \tag{71}$$

where

$$\mathbf{Q}_\theta^{-1}\{ij\} = \begin{cases} \frac{p_{\theta i}+p_{\theta(i+1)}}{p_{\theta i}p_{\theta(i+1)}} & i = j \\ -\frac{1}{p_{\theta i}} & i = j+1 \\ -\frac{1}{p_{\theta j}} & j = i+1 \\ 0 & \text{otherwise} \end{cases} \tag{72}$$

$\mathbf{Q}_\theta^{-1} \in \mathbb{R}^{m\times m}$ has $3m$ nonzero entries only, whose decomposition can be processed in $\mathcal{O}(m)$ steps. After decomposition, we have $\mathbf{Q}_\theta^{-1} = \mathbf{F}_\theta^{-1}(\mathbf{F}_\theta^{-1})^T$, whose transpose is the target matrix $\mathbf{F}_\theta^{-1}$, and then the overall time complexity is decreased to $\mathcal{O}(nm)$ as well.

6. The Estimation of Distribution

The prerequisite of our algorithm is to know the distribution. In Experiment 2, we didn't mention how to acquire the distributions and assumed we already knew. In this section, we will introduce how to estimate distributions by the median filter, why we choose that, and post-processing steps.

6.1. Distribution Estimation based on Median Filter and Reasons

In some cases, we may have prior knowledge to derive the distribution, based on the noise model and camera system parameters. Sometimes we may obtain the distribution accurately if it's possible to take multiple static images. Otherwise, we have to estimate the distribution from the data.

We recommend to estimate distributions from bioimaging data by median filter for three reasons. First of all, it's fast enough. Many blind noise level estimation algorithms are based on matrix decomposition or principal component analysis, such as (Pyatykh et al., 2012), (Chen et al., 2015) and (Liu et al., 2013). Other methods use different operations like grouping similar blocks (Dabov et al., 2007). However, all these methods are too time-consuming to be applied to large-size bioimaging data.

What's more, bioimages usually have a lower signal-to-noise ratio, blurrier boundaries, and smoother gradients compared to natural images, where is more reasonable to assume the neighborhood pixels have similar signals and distributions. Then, we can infer the signal θ behind an observation based on the information of its local patch. The noise distribution can be approximately estimated by counting all pixels with the same signal θ.

It's natural to use the mean of all observations to estimate the signal if only the first two reasons are given. However, as mentioned before, the expectation of a r.v. may not equal to the signal. The third reason to use a median filter is that clipping and nonlinear transformation can only change the value information but the order information is still kept. For the Poisson distribution $\mathcal{P}(\theta)$, the lower bound and the upper bound of the median is $\theta - 2/3$ and $\theta + 1/3$ (Adell & Jodrá, 2005). When θ are all integers, we have $Median(X_\theta) = \theta$ exactly so that the MSE estimated by the median is the same as the real one.

6.2. Truncated Gaussian Model Fitting

The estimation method in 6.1 is based on the assumption that the neighborhood pixels have similar signals. Some pixels on corners or edges do not satisfy the assumption and should be removed. Here we propose an option to improve the accuracy of estimated variances (or MSEs) when distributions are rough.

For the pixels dissatisfying the assumption, the estimated errors are usually much larger than the real noise so that these pixels concentrate in the tails of distributions. If we truncate the tails, the remaining part of distributions can be more accurate. However, the tails contribute most of the variance and cannot be ignored. To recover the real variance, we regard the noise distribution before truncation as Gaussian distribution. If a r.v. X follows the standard Gaussian distribution and truncated between α and β, we have (Barr & Sherrill, 1999):

$$Var[X] = \frac{Var[X|\alpha < X < \beta]}{1 + \frac{\alpha\phi(\alpha)-\beta\phi(\beta)}{\Phi(\beta)-\Phi(\alpha)} - (\frac{\alpha\phi(\alpha)-\beta\phi(\beta)}{\Phi(\beta)-\Phi(\alpha)})^2}, \tag{73}$$

where ϕ and Φ are the probability density function and the cumulative distribution function of the standard Gaussian distribution. The variance of the truncated distribution is proportional to the original variance. When α and β are chosen, the denominator of Equation 73 is a factor to recover the real variance labeled as t_θ. The optimization problem can be reformulated as $C'_f = \max_\theta \Delta\mathbf{x}^T\mathbf{Q}_\theta\Delta\mathbf{x}/t_\theta$. As a heuristic rule of thumb, we recommend truncating 3% of all samples.

7. Computational Time Comparison

The two peer methods have superior time complexities. Although the time complexity of GAT is $\mathcal{O}(m)$ because all transformations need to be calculated once, it's just an analytical function almost without computation, whose execution time is negligible. Foi's method has $\mathcal{O}(kn)$ time complexity, where k is the number of iterations.

In this paper, we only compared 8-bit image data, where all methods were too quick to be meaningfully distinguished. To compare the running time, we did a new experiment on clipped Poissonian-Gaussian models with size of Θs ranging from 2^8 to 2^{12}. GAT is not compared, it's still very quick even for the largest size. Foi's method was iterated 500 times, and only the first part of ConvexOpt was executed.

All experiments were executed on a workstation Dell® Precision 7920 Tower. The CPU is Intel® Xeon® Gold 6140, whose clock rate is 2.30 GHz. The memory size is 128GB (up to 2GB was used). No GPU was used. The operating system is Windows 10, and all codes are MATLAB, whose version is R2018b. MOSEK optimizer was used, whose version is 9.0.

Table 1. Computational Time

Algorithm	$n = 2^8$	$n = 2^9$	$n = 2^{10}$	$n = 2^{11}$	$n = 2^{12}$
ConvexOpt	4.92	23.6	110	519	2940
Foi	5.81	20.9	71.6	433	2376

Table 1 shows the computational time of the two methods. When n is not very large, they are comparable. However, ConvexOpt is worse with a bigger size of Θ.

References

Adell, J. A. and Jodrá, P. The median of the poisson distribution. *Metrika*, 61(3):337–346, 2005.

Andersen, E. D., Roos, C., and Terlaky, T. On implementing a primal-dual interior-point method for conic quadratic optimization. *Mathematical Programming*, 95(2):249–277, 2003.

Barr, D. R. and Sherrill, E. T. Mean and variance of truncated normal distributions. *The American Statistician*, 53(4): 357–361, 1999.

Boyd, S. and Vandenberghe, L. *Convex optimization*. Cambridge university press, 2004.

Chen, G., Zhu, F., and Ann Heng, P. An efficient statistical method for image noise level estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 477–485, 2015.

Dabov, K., Foi, A., Katkovnik, V., and Egiazarian, K. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on image processing*, 16(8):2080–2095, 2007.

Liu, X., Tanaka, M., and Okutomi, M. Single-image noise level estimation for blind denoising. *IEEE transactions on image processing*, 22(12):5226–5237, 2013.

Nesterov, Y. E. and Todd, M. J. Self-scaled barriers and interior-point methods for convex programming. *Mathematics of Operations research*, 22(1):1–42, 1997.

Pyatykh, S., Hesser, J., and Zheng, L. Image noise level estimation by principal component analysis. *IEEE transactions on image processing*, 22(2):687–699, 2012.