Consider the linearly-constrained convex programming problem

\[ \Phi^* = \text{Minimum } F(x) \]
\[ \text{subject to } Ax = b \]
\[ x \geq 0 \]

where \( F \) is a convex function.

Barrier function:

\[ \Phi(\zeta) = \text{Minimum } F(x) - \zeta \sum_{j=1}^{n} \ln x_j \]
\[ \text{subject to } Ax = b, \quad x \geq 0 \]
\[ \text{as } x \to 0, \quad -\zeta \ln(x) \to \infty \]

so the minimizer \( x(\zeta) \) of \( \Phi(\zeta) \) is positive!

Lagrangian function:

\[ L(x, y, \zeta) = F(x) - \zeta \sum_{j=1}^{n} x_j - y^T(Ax - b) \]

Optimality conditions for \( \Phi(\zeta) \):

\[ \frac{\partial L(x, y, \zeta)}{\partial x_j} = 0, \quad j = 1, \ldots, n \]
\[ \frac{\partial L(x, y, \zeta)}{\partial y_i} = 0, \quad i = 1, \ldots, m \]
\[ x_j \left( \frac{\partial L(x, y, \zeta)}{\partial x_j} \right) = 0 \]

Notation:

\[ e^T = (1, 1, 1, \ldots, 1) \]

\[ X = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \]

\[ \Rightarrow X^{-1} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \]

\[ \frac{\partial L(x, y, \zeta)}{\partial x_j} = 0, \quad j = 1, \ldots, n \]
\[ \Rightarrow \nabla F(x) - A^T y - \zeta X^{-1} e \geq 0 \]

The minimizer is positive, i.e., \( x(\zeta) > 0 \)

so that it must satisfy

\[ \nabla F(x) - A^T y - \zeta X^{-1} e = 0 \]

together with

\[ Ax = b \]
Optimality conditions for $\Phi(\zeta)$ are
\[
\begin{cases}
\nabla F(x) \cdot A^T y - \zeta x^{-1} e = 0 \\
A x = b
\end{cases}
\]
Define
\[ s = \nabla F(x) - yA^T \]
so that the first condition is
\[ s - \zeta x^{-1} e = 0 \]
\[ \Rightarrow \begin{cases} 
X \ s = \zeta e \\
A \ x = b
\end{cases} \]

We want to use an iteration of the Newton-Raphson method to solve the nonlinear system
\[
\begin{cases}
X \ s = \zeta e \\
A \ x = b \\
s = \nabla F(x) - A^T y
\end{cases}
\]

Since $x^k > 0$, $X^k$ is nonsingular, and
\[
s^k + \nabla^2 F(x^k) \Delta x - A^T \Delta y + [X^k]^{-1} S^k \Delta x = \zeta [X^k]^{-1} e
\]

Newton-Raphson step is found by solving the linear system:
\[
\begin{cases}
\nabla^2 F(x^k) + (X^k)^{-1} S^k \Delta x - A^T \Delta y = \zeta (X^k)^{-1} e - s^k \\
A \Delta x = 0
\end{cases}
\]
and
\[
\begin{cases}
x^{k+1} = x^k + \Delta x \\
y^{k+1} = y^k + \Delta y \\
S^{k+1} = \nabla F(x^{k+1}) - A^T y^{k+1} \\
Z^{k+1} = 1 \frac{1}{n} x^{k+1} s^{k+1}
\end{cases}
\]

If we reduce the barrier parameter by
\[ \tilde{\zeta} = \omega \zeta \]
for some factor $0 < \omega < 1$
then we can perform another Newton-Raphson step to solve (approximately) the new nonlinear system of equations:
\[
\begin{cases}
X \ s = \tilde{\zeta} e \\
A \ x = b
\end{cases}
\]

**Geometric Programming Primal Problem**
\[
\begin{align*}
\text{Minimize} & \quad g_0(t) \\
\text{subject to} & \quad g_j(t) \leq 1, \ k=1,\cdots,K \\
& \quad t_i > 0, \ i=1,2,\cdots,N
\end{align*}
\]
where
\[
g_0(t) = \sum_{j \in [k]} c_j \prod_{i=1}^{N} \tilde{z}_j \\
c_j > 0
\]
\[ \bigcup_{k} [k] = \{1,2,\cdots,N\} \quad \& \quad [k] \cap [k^*] = \emptyset \quad \text{for} \quad k^* \neq k^* \]
Application to Geometric Programming Dual Problem

The GP dual problem is linearly-constrained, and (if the negative of the log of the objective is minimized) has a convex objective function.

$$\text{Maximize } v(\delta, \lambda) = \prod_{k=0}^{K} \left( \lambda_k \right) \prod_{i \in [k]} \left( c_i \right)$$

is equivalent to

$$\text{Max } -\ln v(\delta, \lambda) = \sum_{i=1}^{N} \left( \delta_i \ln c_i - \delta_i \ln \delta_i \right) + \sum_{k=0}^{K} \lambda_k \ln \lambda_k$$

or

$$\text{Min } -\ln v(\delta, \lambda) = \sum_{i=1}^{N} \left( \delta_i \ln \delta_i - \delta_i \ln c_i \right) - \sum_{k=0}^{K} \lambda_k \ln \lambda_k$$

DGP: $$\text{Maximize } v(\delta, \lambda) = \prod_{k=0}^{K} \left( \lambda_k \right) \prod_{i \in [k]} \left( c_i \right) \delta_i$$

subject to

$$\sum_{i \in [k]} \delta_i = \lambda_k, \quad k=0,1,\cdots, K$$

$$\sum_{j=1}^{N} a_{ij} \delta_i = 0, \quad j=1,\cdots, M$$

$$\lambda_k = 1$$

$$\delta_i \geq 0, \lambda_k \geq 0 \quad \forall i, k$$

Note: $$\bigcup_{k} [k] = \{1,2,\cdots,N\} \& \{k\} \cap \{k'\} = \emptyset \quad \text{for } k' \neq k'$$

DGP': $$\text{Min } \sum_{i=1}^{N} \left( \delta_i \ln \delta_i - \delta_i \ln c_i \right) - \sum_{k=0}^{K} \lambda_k \ln \lambda_k$$

subject to

$$\sum_{i \in [k]} \delta_i = 1 \quad \text{normality}$$

$$\sum_{j=1}^{N} a_{ij} \delta_i = 0, \quad j=1,\cdots, M$$

$$\delta_i \geq 0, \quad \forall i$$

DGP' has several noteworthy properties:

- objective is convex
- constraints are linear
- if primal constraint $k$ is slack
  $$\Rightarrow \lambda_k = 0 = \sum_{i \in [k]} \delta_i = 0 \quad \Rightarrow \delta_i = 0 \quad \forall i \in [k]$$

The terms $\delta \ln \delta$ are difficult to compute for small positive $\delta$.

While we may define

$$\lim_{\delta \to 0} \delta \ln \delta = 0$$

$\delta \ln \delta$ is not differentiable at 0.

The objective is infinitely differentiable at positive $\delta$.

The algorithm which we have described was implemented in an experimental APL code PFAGP1.

$$x \leftrightarrow \delta$$

$$y \leftrightarrow \ln t$$

$$F(x) \leftrightarrow -\ln V(\delta)$$

The primal GP variables $(t)$ are obtained by exponentiating $y$. 
Obtaining the starting solution \((x^0, s^0, y^0)\)

\[
\begin{align*}
X^0 & s^0 = \zeta^0 e \\
A x^0 & = b \\
s^0 & = VF(s^0) - A^T y^0
\end{align*}
\]

is accomplished by arbitrarily choosing \(x\) & \(y\), & adding an artificial variable \(x_{N+1}\) with large cost, & a bounding constraint \(\sum_{i=1}^{N+1} x_i \leq U\) with large \(U\)

setting \(\zeta^0 = \frac{1}{N+1} \sum_{i=1}^{N+1} x_i s_i^0\)

As the exponent \(\alpha\) is varied from \(-\frac{1}{\epsilon_3}\) to \(+1\), the first & last constraints change from being both slack to both active.

The GP dual problem is rather badly conditioned. For example, in the case \(\alpha=+1\), the optimal values of three of the positive dual variables are less than \(0.001\), and most dual-based GP algorithms experience difficulty in determining the optimal primal variables.

### Computational Experience

Beck & Ecker's Problem #10 will be used to illustrate the behavior of this algorithm

- **# variables** = 7
- **# constraints** = 4
- **# terms** = 20
- **"degrees of difficulty"** = 12

There are 6 different variations, in which the exponent \(\alpha\) in one term is varied.

<table>
<thead>
<tr>
<th>Iteration</th>
<th>(x)</th>
<th>(y)</th>
<th>(z/2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5.5659774E1</td>
<td>2.0663538E15</td>
<td>1.4935504E15</td>
</tr>
<tr>
<td>2</td>
<td>4.8741341E1</td>
<td>2.0983771E15</td>
<td>1.4935504E15</td>
</tr>
<tr>
<td>3</td>
<td>2.6933823E1</td>
<td>2.0289012E15</td>
<td>1.4935504E15</td>
</tr>
<tr>
<td>4</td>
<td>2.0504211E1</td>
<td>4.1120821E15</td>
<td>1.4935504E15</td>
</tr>
<tr>
<td>5</td>
<td>0.2967922E16</td>
<td>2.0962206E15</td>
<td>1.4935504E15</td>
</tr>
<tr>
<td>6</td>
<td>6.1679218E15</td>
<td>2.0962206E15</td>
<td>1.4935504E15</td>
</tr>
<tr>
<td>7</td>
<td>1.0920453E15</td>
<td>2.0962206E15</td>
<td>1.4935504E15</td>
</tr>
<tr>
<td>8</td>
<td>0.2967922E16</td>
<td>2.0962206E15</td>
<td>1.4935504E15</td>
</tr>
<tr>
<td>9</td>
<td>1.3891288E16</td>
<td>2.0962206E15</td>
<td>1.4935504E15</td>
</tr>
<tr>
<td>10</td>
<td>1.3891288E16</td>
<td>2.0962206E15</td>
<td>1.4935504E15</td>
</tr>
</tbody>
</table>

**Objective function** = 1.5000522  
**Log 10** = 7.9054561  
**Log** = 0.75892885 **Iter** = 16, 1.5939050E15

### Dual Variables (of the dual GP problem) \(y\):

<table>
<thead>
<tr>
<th>Iteration</th>
<th>(y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7.9054561E15</td>
</tr>
<tr>
<td>2</td>
<td>6.1802187E15</td>
</tr>
<tr>
<td>3</td>
<td>2.4107774E15</td>
</tr>
<tr>
<td>4</td>
<td>4.7107774E15</td>
</tr>
<tr>
<td>5</td>
<td>6.8947774E15</td>
</tr>
<tr>
<td>6</td>
<td>7.9054561E15</td>
</tr>
<tr>
<td>7</td>
<td>7.4805678E15</td>
</tr>
<tr>
<td>8</td>
<td>1.4873678E15</td>
</tr>
</tbody>
</table>

### Constraints

<table>
<thead>
<tr>
<th><strong>#</strong></th>
<th><strong>Value</strong></th>
<th><strong>Infeasibility</strong></th>
<th><strong>Infeasibility</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.00000000002</td>
<td>0.00000000002</td>
<td>0.00000000002</td>
</tr>
<tr>
<td>3</td>
<td>0.00000000000</td>
<td>0.00000000000</td>
<td>0.00000000000</td>
</tr>
<tr>
<td>4</td>
<td>0.00000000000</td>
<td>0.00000000000</td>
<td>0.00000000000</td>
</tr>
<tr>
<td>5</td>
<td>0.00000000000</td>
<td>0.00000000000</td>
<td>0.00000000000</td>
</tr>
</tbody>
</table>

**Log 10** \(\zeta\)  
**Iter** = 16, 1.5939050E15
These two variables have optimal
values which are small, but positive:
$\delta_3 = 0.001226$
$\delta_5 = 0.001513$

...looking at the final iterates
$\delta_3^* = 0.001226$
$\delta_5^* = 0.001513$

Dual-based GP algorithms generally have difficulty with problems such as the following:

Minimize $g_0(t) = t_1t_2 + t_1^{1/4}t_2^{1/2}$
subject to
$g_2(t) = 1 - t_1^{0.5} + t_2 \leq 1$
$t_1 > 0$, $t_2 > 0$
The dual problem has a unique solution
\[ \delta_1^* = \delta_2^* = 1/2, \quad \lambda_1 = \delta_3^* = \delta_4^* = 0 \]

The primal solution, which is not unique, cannot be determined by solving the equations
\[ c_j \prod_{i=1}^{k} x_i^{\delta_j} = \frac{\delta_j}{\lambda_k} \quad \text{where} \quad j \in [k] \]

\[ g_1(t) \]
\[ 0 \quad 10 \quad 20 \quad 30 \]

**The algorithm, because dual variables are kept positive, converges to a primal optimal solution**

Max: \[ \ln(\delta, \lambda) = \sum_{i=1}^{N} \{ \delta_i \ln c_i - \delta_i \ln \delta_i \} + \sum_{k=1}^{K} \lambda_k \ln \lambda_k \]
subject to
\[ \sum_{i=1}^{N} \delta_i = \lambda_k, \quad k = 0, 1, \ldots, K \]
\[ \sum_{j=1}^{M} a_{ij} \delta_i = 0, \quad j = 1, \ldots, M \]
\[ \delta_i \geq 0, \lambda_k \geq 0 \]

Geometric Programming Dual Problem

Make a change of variable:
\[ \delta_j = \rho_j \lambda_k \quad \text{for} \quad j \in [k] \]
so that \[ \rho_j = \frac{\delta_j}{\lambda_k} \quad \text{if} \quad j \in [k] \quad \text{and} \quad \lambda_k > 0 \]

Define functions
\[ G_k(\rho) = \sum_{j \in [k]} \left[ \rho_j \ln c_j - \rho_j \ln \rho_j \right] \quad \text{entropy function} \]
\[ A_k(\rho) = \sum_{j \in [k]} a_{ij} \rho_j \]
Maximize \( \sum_{k=0}^{p} G_k(p) \lambda_k \)
subject to \( \sum_{k=0}^{p} A_{ki}(p) \lambda_k = 0, \quad i=1, \ldots, N \)
\[ \lambda_0 = 1 \]
\[ \sum_{j \in [k]} \rho_j = 1, \quad k=0,1,\ldots, K \]
\[ \lambda_k \geq 0, \quad \rho_j \geq 0, \quad \forall k,j \]

For fixed values of \( \lambda \), this is an entropy problem.

For fixed values of \( p \), this is an LP problem.

The path-following algorithm has polynomial-time complexity for both entropy & LP problems.

If \( t \) is optimal in the primal, and \( (p, \lambda) \) is optimal in the dual, then
\[ \rho_j > 0 \] and \( g_i(t) > 0 \)
and
\[ \rho_j = \frac{\sum_{i=1}^{N} \prod_{l=1}^{i} t_i^{\rho_l}}{g_i(t)} \]

whether the constraint \( k \) is tight or slack.