An Interior–Point Algorithm for Nonconvex Nonlinear Programming

Robert J. Vanderbei

joint work with Hande Y. Benson and David F. Shanno

July 10, 2000

Level(3) ^{SM} Communications

Operations Research and Financial Engineering, Princeton University

http://www.princeton.edu/∼rvdb

1 Outline

- The Basic Interior-Point Paradigm (for LP/QP)
- Modifications for Convex Optimization
- Modifications for Nonconvex Optimization
- Computational Results

1 Outline

- The Basic Interior-Point Paradigm (for LP/QP)
- Modifications for Convex Optimization
- Modifications for Nonconvex Optimization
- Computational Results

1 Outline

- The Basic Interior-Point Paradigm (for LP/QP)
- Modifications for Convex Optimization
- Modifications for Nonconvex Optimization
- Computational Results

1 Outline

- The Basic Interior-Point Paradigm (for LP/QP)
- Modifications for Convex Optimization
- Modifications for Nonconvex Optimization
- Computational Results

1 Outline

- The Basic Interior-Point Paradigm (for LP/QP)
- Modifications for Convex Optimization
- Modifications for Nonconvex Optimization
- Computational Results

The Basic Interior-Point Paradigm (for LP/QP)

2 Introduce Slack Variables

- Start with an optimization problem—for now, the simplest NLP:

$$\begin{aligned} &\text{minimize} && f(x) \\ &\text{subject to} && h_i(x) \geq 0, \qquad i = 1, \ldots, m \end{aligned}$$

- Introduce slack variables to make all inequality constraints into nonnegativities:

$$\begin{aligned} &\text{minimize} && f(x) \\ &\text{subject to} && h(x) - w = 0, \\ & && w \geq 0 \end{aligned}$$

2 Introduce Slack Variables

- Start with an optimization problem—for now, the simplest NLP:

$$\begin{aligned} &\text{minimize} && f(x) \\ &\text{subject to} && h_i(x) \geq 0, \qquad i = 1, \ldots, m \end{aligned}$$

- Introduce slack variables to make all inequality constraints into nonnegativities:

$$\begin{aligned} &\text{minimize} && f(x) \\ &\text{subject to} && h(x) - w = 0, \\ & && w \geq 0 \end{aligned}$$

2 Introduce Slack Variables

- Start with an optimization problem—for now, the simplest NLP:

$$\begin{aligned} &\text{minimize} && f(x) \\ &\text{subject to} && h_i(x) \geq 0, \qquad i = 1, \ldots, m \end{aligned}$$

- Introduce slack variables to make all inequality constraints into nonnegativities:

$$\begin{aligned} &\text{minimize} && f(x) \\ &\text{subject to} && h(x) - w = 0, \\ & && w \geq 0 \end{aligned}$$

3 Associated Log-Barrier Problem

- Replace nonnegativity constraints with logarithmic barrier terms in the objective:

$$\begin{aligned} \text{minimize} \quad & f(x) - \mu \sum_{i=1}^{m} \log(w_i) \\ \text{subject to} \quad & h(x) - w = 0 \end{aligned}$$

3 Associated Log-Barrier Problem

- Replace nonnegativity constraints with logarithmic barrier terms in the objective:

$$\begin{aligned} \text{minimize} \quad & f(x) - \mu \sum_{i=1}^{m} \log(w_i) \\ \text{subject to} \quad & h(x) - w = 0 \end{aligned}$$

First-Order Optimality Conditions

- Incorporate the equality constraints into the objective using Lagrange multipliers:

$$L(x, w, y) = f(x) - \mu \sum_{i=1}^{m} \log(w_i) - y^T(h(x) - w)$$

- Set all derivatives to zero:

$$\begin{aligned} \nabla f(x) - \nabla h(x)^T y &= 0 \\ -\mu W^{-1} e + y &= 0 \\ h(x) - w &= 0 \end{aligned}$$

First-Order Optimality Conditions

- Incorporate the equality constraints into the objective using Lagrange multipliers:

$$L(x, w, y) = f(x) - \mu \sum_{i=1}^{m} \log(w_i) - y^T(h(x) - w)$$

- Set all derivatives to zero:

$$\begin{aligned} \nabla f(x) - \nabla h(x)^T y &= 0 \\ -\mu W^{-1} e + y &= 0 \\ h(x) - w &= 0 \end{aligned}$$

First-Order Optimality Conditions

- Incorporate the equality constraints into the objective using Lagrange multipliers:

$$L(x, w, y) = f(x) - \mu \sum_{i=1}^{m} \log(w_i) - y^T(h(x) - w)$$

- Set all derivatives to zero:

$$\begin{aligned} \nabla f(x) - \nabla h(x)^T y &= 0 \\ -\mu W^{-1} e + y &= 0 \\ h(x) - w &= 0 \end{aligned}$$

5 Symmetrize Complementarity Conditions

- Rewrite system:

$$\begin{aligned} \nabla f(x) - \nabla h(x)^T y &= 0 \\ WYe &= \mu e \\ h(x) - w &= 0 \end{aligned}$$

5 Symmetrize Complementarity Conditions

- Rewrite system:

$$\begin{aligned} \nabla f(x) - \nabla h(x)^T y &= 0 \\ WYe &= \mu e \\ h(x) - w &= 0 \end{aligned}$$

6 Apply Newton's Method

- Apply Newton's method to compute search directions, Δx, Δw, Δy:

$$\begin{bmatrix} H(x,y) & 0 & -A(x)^T \\ 0 & Y & W \\ A(x) & -I & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta w \\ \Delta y \end{bmatrix} = \begin{bmatrix} -\nabla f(x) + A(x)^T y \\ \mu e - WYe \\ -h(x) + w \end{bmatrix}.$$

  Here,

$$H(x,y) = \nabla^2 f(x) - \sum_{i=1}^{m} y_i \nabla^2 h_i(x)$$

  and

$$A(x) = \nabla h(x)$$

- Note: $H(x,y)$ is positive semidefinite if f is convex, each h_i is concave, and each $y_i \geq 0$.

6 Apply Newton's Method

- Apply Newton's method to compute search directions, Δx, Δw, Δy:

$$\begin{bmatrix} H(x,y) & 0 & -A(x)^T \\ 0 & Y & W \\ A(x) & -I & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta w \\ \Delta y \end{bmatrix} = \begin{bmatrix} -\nabla f(x) + A(x)^T y \\ \mu e - WYe \\ -h(x) + w \end{bmatrix}.$$

Here,

$$H(x,y) = \nabla^2 f(x) - \sum_{i=1}^{m} y_i \nabla^2 h_i(x)$$

and

$$A(x) = \nabla h(x)$$

- Note: $H(x,y)$ is positive semidefinite if f is convex, each h_i is concave, and each $y_i \geq 0$.

6 Apply Newton's Method

- Apply Newton's method to compute search directions, Δx, Δw, Δy:

$$\begin{bmatrix} H(x,y) & 0 & -A(x)^T \\ 0 & Y & W \\ A(x) & -I & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta w \\ \Delta y \end{bmatrix} = \begin{bmatrix} -\nabla f(x) + A(x)^T y \\ \mu e - WYe \\ -h(x) + w \end{bmatrix}.$$

Here,

$$H(x,y) = \nabla^2 f(x) - \sum_{i=1}^{m} y_i \nabla^2 h_i(x)$$

and

$$A(x) = \nabla h(x)$$

- Note: $H(x,y)$ is positive semidefinite if f is convex, each h_i is concave, and each $y_i \geq 0$.

6 Apply Newton's Method

- Apply Newton's method to compute search directions, Δx, Δw, Δy:

$$\begin{bmatrix} H(x,y) & 0 & -A(x)^T \\ 0 & Y & W \\ A(x) & -I & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta w \\ \Delta y \end{bmatrix} = \begin{bmatrix} -\nabla f(x) + A(x)^T y \\ \mu e - WYe \\ -h(x) + w \end{bmatrix}.$$

Here,

$$H(x,y) = \nabla^2 f(x) - \sum_{i=1}^{m} y_i \nabla^2 h_i(x)$$

and

$$A(x) = \nabla h(x)$$

- Note: $H(x,y)$ is positive semidefinite if f is convex, each h_i is concave, and each $y_i \geq 0$.

6 Apply Newton's Method

- Apply Newton's method to compute search directions, Δx, Δw, Δy:

$$\begin{bmatrix} H(x,y) & 0 & -A(x)^T \\ 0 & Y & W \\ A(x) & -I & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta w \\ \Delta y \end{bmatrix} = \begin{bmatrix} -\nabla f(x) + A(x)^T y \\ \mu e - WYe \\ -h(x) + w \end{bmatrix}.$$

  Here,

$$H(x,y) = \nabla^2 f(x) - \sum_{i=1}^{m} y_i \nabla^2 h_i(x)$$

  and

$$A(x) = \nabla h(x)$$

- Note: $H(x,y)$ is positive semidefinite if f is convex, each h_i is concave, and each $y_i \geq 0$.

7 Reduced KKT System

- Use second equation to solve for Δw. Result is the reduced KKT system:

$$\begin{bmatrix} -H(x,y) & A^T(x) \\ A(x) & WY^{-1} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} \nabla f(x) - A^T(x)y \\ -h(x) + \mu Y^{-1} e \end{bmatrix}$$

- Iterate:

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)} \Delta x^{(k)}$$
$$w^{(k+1)} = w^{(k)} + \alpha^{(k)} \Delta w^{(k)}$$
$$y^{(k+1)} = y^{(k)} + \alpha^{(k)} \Delta y^{(k)}$$

7 Reduced KKT System

- Use second equation to solve for Δw. Result is the reduced KKT system:

$$\begin{bmatrix} -H(x,y) & A^T(x) \\ A(x) & WY^{-1} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} \nabla f(x) - A^T(x)y \\ -h(x) + \mu Y^{-1} e \end{bmatrix}$$

- Iterate:

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)} \Delta x^{(k)}$$
$$w^{(k+1)} = w^{(k)} + \alpha^{(k)} \Delta w^{(k)}$$
$$y^{(k+1)} = y^{(k)} + \alpha^{(k)} \Delta y^{(k)}$$

7 Reduced KKT System

- Use second equation to solve for Δw. Result is the reduced KKT system:

$$\begin{bmatrix} -H(x,y) & A^T(x) \\ A(x) & WY^{-1} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} \nabla f(x) - A^T(x)y \\ -h(x) + \mu Y^{-1} e \end{bmatrix}$$

- Iterate:

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)} \Delta x^{(k)}$$
$$w^{(k+1)} = w^{(k)} + \alpha^{(k)} \Delta w^{(k)}$$
$$y^{(k+1)} = y^{(k)} + \alpha^{(k)} \Delta y^{(k)}$$

Modifications for Convex Optimization

8 Need to Use Shorter Steps

For convex nonquadratic optimization, it does not suffice to choose the steplength α simply to maintain positivity of nonnegative variables.

- Consider, e.g., minimizing

$$f(x) = (1 + x^2)^{1/2}.$$

- The iterates can be computed explicitly:

$$x^{(k+1)} = -(x^{(k)})^3$$

- Converges if and only if $|x| \leq 1$.
- Reason: away from 0, function is too linear.

8 Need to Use Shorter Steps

For convex nonquadratic optimization, it does not suffice to choose the steplength α simply to maintain positivity of nonnegative variables.

- Consider, e.g., minimizing

$$f(x) = (1 + x^2)^{1/2}.$$

- The iterates can be computed explicitly:

$$x^{(k+1)} = -(x^{(k)})^3$$

- Converges if and only if $|x| \leq 1$.
- Reason: away from 0, function is too linear.

8 Need to Use Shorter Steps

For convex nonquadratic optimization, it does not suffice to choose the steplength α simply to maintain positivity of nonnegative variables.

- Consider, e.g., minimizing

$$f(x) = (1 + x^2)^{1/2}.$$

- The iterates can be computed explicitly:

$$x^{(k+1)} = -(x^{(k)})^3$$

- Converges if and only if $|x| \leq 1$.
- Reason: away from 0, function is too linear.

8 Need to Use Shorter Steps

For convex nonquadratic optimization, it does not suffice to choose the steplength α simply to maintain positivity of nonnegative variables.

- Consider, e.g., minimizing

$$f(x) = (1 + x^2)^{1/2}.$$

- The iterates can be computed explicitly:

$$x^{(k+1)} = -(x^{(k)})^3$$

- Converges if and only if $|x| \leq 1$.
- Reason: away from 0, function is too linear.

8 Need to Use Shorter Steps

For convex nonquadratic optimization, it does not suffice to choose the steplength α simply to maintain positivity of nonnegative variables.

- Consider, e.g., minimizing

$$f(x) = (1 + x^2)^{1/2}.$$

- The iterates can be computed explicitly:

$$x^{(k+1)} = -(x^{(k)})^3$$

- Converges if and only if $|x| \leq 1$.
- Reason: away from 0, function is too linear.

9 Merit Function

A merit function is used to guide the choice of steplength α.

We use the Fiacco–McCormick merit function

$$\Psi_{\beta,\mu}(x,w) = f(x) - \mu \sum_{i=1}^{m} \log(w_i) + \tfrac{\beta}{2}\|h(x) - w\|_2^2.$$

Define the dual normal matrix:

$$N(x,y,w) = H(x,y) + A^T(x) W^{-1} Y A(x).$$

Theorem 1 *Suppose that $N(x,y,w)$ is positive definite.*

1. *For β sufficiently large, $(\Delta x, \Delta w)$ is a descent direction for $\Psi_{\beta,\mu}$.*
2. *If current solution is primal feasible, then $(\Delta x, \Delta w)$ is a descent direction for the barrier function.*

Note: minimum required value for β is easy to compute.

9 Merit Function

A merit function is used to guide the choice of steplength α.

We use the Fiacco–McCormick merit function

$$\Psi_{\beta,\mu}(x,w) = f(x) - \mu \sum_{i=1}^{m} \log(w_i) + \frac{\beta}{2}\|h(x) - w\|_2^2.$$

Define the dual normal matrix:

$$N(x,y,w) = H(x,y) + A^T(x) W^{-1} Y A(x).$$

Theorem 1 *Suppose that $N(x,y,w)$ is positive definite.*

1. *For β sufficiently large, $(\Delta x, \Delta w)$ is a descent direction for $\Psi_{\beta,\mu}$.*
2. *If current solution is primal feasible, then $(\Delta x, \Delta w)$ is a descent direction for the barrier function.*

Note: minimum required value for β is easy to compute.

9 Merit Function

A merit function is used to guide the choice of steplength α.

We use the Fiacco–McCormick merit function

$$\Psi_{\beta,\mu}(x,w) = f(x) - \mu \sum_{i=1}^{m} \log(w_i) + \frac{\beta}{2}\|h(x) - w\|_2^2.$$

Define the dual normal matrix:

$$N(x,y,w) = H(x,y) + A^T(x) W^{-1} Y A(x).$$

Theorem 1 *Suppose that $N(x,y,w)$ is positive definite.*

1. *For β sufficiently large, $(\Delta x, \Delta w)$ is a descent direction for $\Psi_{\beta,\mu}$.*
2. *If current solution is primal feasible, then $(\Delta x, \Delta w)$ is a descent direction for the barrier function.*

Note: minimum required value for β is easy to compute.

9 Merit Function

A merit function is used to guide the choice of steplength α.

We use the Fiacco–McCormick merit function

$$\Psi_{\beta,\mu}(x,w) = f(x) - \mu \sum_{i=1}^{m} \log(w_i) + \frac{\beta}{2}\|h(x) - w\|_2^2.$$

Define the dual normal matrix:

$$N(x,y,w) = H(x,y) + A^T(x) W^{-1} Y A(x).$$

Theorem 1 *Suppose that $N(x,y,w)$ is positive definite.*

1. *For β sufficiently large, $(\Delta x, \Delta w)$ is a descent direction for $\Psi_{\beta,\mu}$.*
2. *If current solution is primal feasible, then $(\Delta x, \Delta w)$ is a descent direction for the barrier function.*

Note: minimum required value for β is easy to compute.

9 Merit Function

A merit function is used to guide the choice of steplength α.

We use the Fiacco–McCormick merit function

$$\Psi_{\beta,\mu}(x,w) = f(x) - \mu \sum_{i=1}^{m} \log(w_i) + \frac{\beta}{2}\|h(x) - w\|_2^2.$$

Define the dual normal matrix:

$$N(x,y,w) = H(x,y) + A^T(x) W^{-1} Y A(x).$$

Theorem 1 *Suppose that $N(x,y,w)$ is positive definite.*

1. *For β sufficiently large, $(\Delta x, \Delta w)$ is a descent direction for $\Psi_{\beta,\mu}$.*
2. *If current solution is primal feasible, then $(\Delta x, \Delta w)$ is a descent direction for the barrier function.*

Note: minimum required value for β is easy to compute.

9 Merit Function

A merit function is used to guide the choice of steplength α.

We use the Fiacco–McCormick merit function

$$\Psi_{\beta,\mu}(x,w) = f(x) - \mu \sum_{i=1}^{m} \log(w_i) + \frac{\beta}{2}\|h(x) - w\|_2^2.$$

Define the dual normal matrix:

$$N(x,y,w) = H(x,y) + A^T(x) W^{-1} Y A(x).$$

Theorem 1 *Suppose that $N(x,y,w)$ is positive definite.*

1. *For β sufficiently large, $(\Delta x, \Delta w)$ is a descent direction for $\Psi_{\beta,\mu}$.*
2. *If current solution is primal feasible, then $(\Delta x, \Delta w)$ is a descent direction for the barrier function.*

Note: minimum required value for β is easy to compute.

9 Merit Function

A merit function is used to guide the choice of steplength α.

We use the Fiacco–McCormick merit function

$$\Psi_{\beta,\mu}(x,w) = f(x) - \mu \sum_{i=1}^{m} \log(w_i) + \frac{\beta}{2}\|h(x) - w\|_2^2.$$

Define the dual normal matrix:

$$N(x,y,w) = H(x,y) + A^T(x) W^{-1} Y A(x).$$

Theorem 1 *Suppose that $N(x,y,w)$ is positive definite.*

1. *For β sufficiently large, $(\Delta x, \Delta w)$ is a descent direction for $\Psi_{\beta,\mu}$.*
2. *If current solution is primal feasible, then $(\Delta x, \Delta w)$ is a descent direction for the barrier function.*

Note: minimum required value for β is easy to compute.

9 Merit Function

A merit function is used to guide the choice of steplength α.

We use the Fiacco–McCormick merit function

$$\Psi_{\beta,\mu}(x,w) = f(x) - \mu \sum_{i=1}^{m} \log(w_i) + \frac{\beta}{2}\|h(x) - w\|_2^2.$$

Define the dual normal matrix:

$$N(x,y,w) = H(x,y) + A^T(x) W^{-1} Y A(x).$$

Theorem 1 *Suppose that $N(x,y,w)$ is positive definite.*

1. *For β sufficiently large, $(\Delta x, \Delta w)$ is a descent direction for $\Psi_{\beta,\mu}$.*
2. *If current solution is primal feasible, then $(\Delta x, \Delta w)$ is a descent direction for the barrier function.*

Note: minimum required value for β is easy to compute.

Modifications for NonConvex Optimization

Nonconvex Optimization: Diagonal Perturbation

- If $H(x, y)$ is not positive semidefinite then $N(x, y, w)$ might fail to be positive definite.
- In such a case, we lose the descent properties given in previous theorem.
- To regain those properties, we perturb the Hessian: $\tilde{H}(x, y) = H(x, y) + \lambda I$.
- And compute search directions using $\tilde{H}$ instead of H.

Notation: let $\tilde{N}$ denote the dual normal matrix associated with $\tilde{H}$.

Theorem 2 *If $\tilde{N}$ is positive definite, then $(\Delta x, \Delta w, \Delta y)$ is a descent direction for*

1. *the primal infeasibility, $\|h(x) - w\|$;*
2. *the noncomplementarity, $w^T y$.*

10 Nonconvex Optimization: Diagonal Perturbation

- If $H(x, y)$ is not positive semidefinite then $N(x, y, w)$ might fail to be positive definite.
- In such a case, we lose the descent properties given in previous theorem.
- To regain those properties, we perturb the Hessian: $\tilde{H}(x, y) = H(x, y) + \lambda I$.
- And compute search directions using $\tilde{H}$ instead of H.

Notation: let $\tilde{N}$ denote the dual normal matrix associated with $\tilde{H}$.

Theorem 2 *If $\tilde{N}$ is positive definite, then $(\Delta x, \Delta w, \Delta y)$ is a descent direction for*

1. *the primal infeasibility, $\|h(x) - w\|$;*
2. *the noncomplementarity, $w^T y$.*

Nonconvex Optimization: Diagonal Perturbation

- If $H(x, y)$ is not positive semidefinite then $N(x, y, w)$ might fail to be positive definite.
- In such a case, we lose the descent properties given in previous theorem.
- To regain those properties, we perturb the Hessian: $\tilde{H}(x, y) = H(x, y) + \lambda I$.
- And compute search directions using $\tilde{H}$ instead of H.

Notation: let $\tilde{N}$ denote the dual normal matrix associated with $\tilde{H}$.

Theorem 2 *If $\tilde{N}$ is positive definite, then $(\Delta x, \Delta w, \Delta y)$ is a descent direction for*

1. *the primal infeasibility, $\|h(x) - w\|$;*
2. *the noncomplementarity, $w^T y$.*

10 Nonconvex Optimization: Diagonal Perturbation

- If $H(x, y)$ is not positive semidefinite then $N(x, y, w)$ might fail to be positive definite.
- In such a case, we lose the descent properties given in previous theorem.
- To regain those properties, we perturb the Hessian: $\tilde{H}(x, y) = H(x, y) + \lambda I$.
- And compute search directions using $\tilde{H}$ instead of H.

Notation: let $\tilde{N}$ denote the dual normal matrix associated with $\tilde{H}$.

Theorem 2 *If $\tilde{N}$ is positive definite, then $(\Delta x, \Delta w, \Delta y)$ is a descent direction for*

1. *the primal infeasibility, $\|h(x) - w\|$;*
2. *the noncomplementarity, $w^T y$.*

10 Nonconvex Optimization: Diagonal Perturbation

- If $H(x, y)$ is not positive semidefinite then $N(x, y, w)$ might fail to be positive definite.
- In such a case, we lose the descent properties given in previous theorem.
- To regain those properties, we perturb the Hessian: $\tilde{H}(x, y) = H(x, y) + \lambda I$.
- And compute search directions using $\tilde{H}$ instead of H.

Notation: let $\tilde{N}$ denote the dual normal matrix associated with $\tilde{H}$.

Theorem 2 *If $\tilde{N}$ is positive definite, then $(\Delta x, \Delta w, \Delta y)$ is a descent direction for*

1. *the primal infeasibility, $\|h(x) - w\|$;*
2. *the noncomplementarity, $w^T y$.*

10 Nonconvex Optimization: Diagonal Perturbation

- If $H(x, y)$ is not positive semidefinite then $N(x, y, w)$ might fail to be positive definite.
- In such a case, we lose the descent properties given in previous theorem.
- To regain those properties, we perturb the Hessian: $\tilde{H}(x, y) = H(x, y) + \lambda I$.
- And compute search directions using $\tilde{H}$ instead of H.

Notation: let $\tilde{N}$ denote the dual normal matrix associated with $\tilde{H}$.

Theorem 2 *If $\tilde{N}$ is positive definite, then $(\Delta x, \Delta w, \Delta y)$ is a descent direction for*

1. *the primal infeasibility, $\|h(x) - w\|$;*
2. *the noncomplementarity, $w^T y$.*

10 Nonconvex Optimization: Diagonal Perturbation

- If $H(x, y)$ is not positive semidefinite then $N(x, y, w)$ might fail to be positive definite.
- In such a case, we lose the descent properties given in previous theorem.
- To regain those properties, we perturb the Hessian: $\tilde{H}(x, y) = H(x, y) + \lambda I$.
- And compute search directions using $\tilde{H}$ instead of H.

Notation: let $\tilde{N}$ denote the dual normal matrix associated with $\tilde{H}$.

Theorem 2 *If $\tilde{N}$ is positive definite, then $(\Delta x, \Delta w, \Delta y)$ is a descent direction for*

1. *the primal infeasibility, $\|h(x) - w\|$;*
2. *the noncomplementarity, $w^T y$.*

10 Nonconvex Optimization: Diagonal Perturbation

- If $H(x, y)$ is not positive semidefinite then $N(x, y, w)$ might fail to be positive definite.
- In such a case, we lose the descent properties given in previous theorem.
- To regain those properties, we perturb the Hessian: $\tilde{H}(x, y) = H(x, y) + \lambda I$.
- And compute search directions using $\tilde{H}$ instead of H.

Notation: let $\tilde{N}$ denote the dual normal matrix associated with $\tilde{H}$.

Theorem 2 *If $\tilde{N}$ is positive definite, then $(\Delta x, \Delta w, \Delta y)$ is a descent direction for*

1. *the primal infeasibility, $\|h(x) - w\|$;*
2. *the noncomplementarity, $w^T y$.*

10 Nonconvex Optimization: Diagonal Perturbation

- If $H(x, y)$ is not positive semidefinite then $N(x, y, w)$ might fail to be positive definite.
- In such a case, we lose the descent properties given in previous theorem.
- To regain those properties, we perturb the Hessian: $\tilde{H}(x, y) = H(x, y) + \lambda I$.
- And compute search directions using $\tilde{H}$ instead of H.

Notation: let $\tilde{N}$ denote the dual normal matrix associated with $\tilde{H}$.

Theorem 2 *If $\tilde{N}$ is positive definite, then $(\Delta x, \Delta w, \Delta y)$ is a descent direction for*

1. *the primal infeasibility, $\|h(x) - w\|$;*
2. *the noncomplementarity, $w^T y$.*

10 Nonconvex Optimization: Diagonal Perturbation

- If $H(x, y)$ is not positive semidefinite then $N(x, y, w)$ might fail to be positive definite.
- In such a case, we lose the descent properties given in previous theorem.
- To regain those properties, we perturb the Hessian: $\tilde{H}(x, y) = H(x, y) + \lambda I$.
- And compute search directions using $\tilde{H}$ instead of H.

Notation: let $\tilde{N}$ denote the dual normal matrix associated with $\tilde{H}$.

Theorem 2 *If $\tilde{N}$ is positive definite, then $(\Delta x, \Delta w, \Delta y)$ is a descent direction for*

1. *the primal infeasibility, $\|h(x) - w\|$;*
2. *the noncomplementarity, $w^T y$.*

11 Notes:

- Not necessarily a descent direction for dual infeasibility.

- A line search is performed to find a value of λ within a factor of 2 of the smallest permissible value.

11 Notes:

- Not necessarily a descent direction for dual infeasibility.

- A line search is performed to find a value of λ within a factor of 2 of the smallest permissible value.

11 Notes:

- Not necessarily a descent direction for dual infeasibility.

- A line search is performed to find a value of λ within a factor of 2 of the smallest permissible value.

Nonconvex Optimization: Jamming

Theorem 3 *If the problem is convex and and the current solution is not optimal and ..., then for any slack variable, say w_i, we have $w_i = 0$ implies $\Delta w_i \geq 0$.*

- To paraphrase: for convex problems, as slack variables get small they tend to get large again. This is an antijamming theorem.
- A recent example of Wächter and Biegler shows that for nonconvex problems, jamming really can occur.
- Recent modification:
  - if a slack variable gets small and
  - its component of the step direction contributes to making a very short step,
  - then increase this slack variable to the average size of the variables the "mainstream" slack variables.
- This modification corrects all examples of jamming that we know about.

Nonconvex Optimization: Jamming

Theorem 3 *If the problem is convex and and the current solution is not optimal and ..., then for any slack variable, say* w_i*, we have* $w_i = 0$ *implies* $\Delta w_i \geq 0$*.*

- To paraphrase: for convex problems, as slack variables get small they tend to get large again. This is an antijamming theorem.
- A recent example of Wächter and Biegler shows that for nonconvex problems, jamming really can occur.
- Recent modification:
  - if a slack variable gets small and
  - its component of the step direction contributes to making a very short step,
  - then increase this slack variable to the average size of the variables the "mainstream" slack variables.
- This modification corrects all examples of jamming that we know about.

12 Nonconvex Optimization: Jamming

Theorem 3 *If the problem is convex and and the current solution is not optimal and ..., then for any slack variable, say* w_i*, we have* $w_i = 0$ *implies* $\Delta w_i \geq 0$*.*

- To paraphrase: for convex problems, as slack variables get small they tend to get large again. This is an antijamming theorem.
- A recent example of Wächter and Biegler shows that for nonconvex problems, jamming really can occur.
- Recent modification:
  - if a slack variable gets small and
  - its component of the step direction contributes to making a very short step,
  - then increase this slack variable to the average size of the variables the "mainstream" slack variables.
- This modification corrects all examples of jamming that we know about.

Nonconvex Optimization: Jamming

Theorem 3 *If the problem is convex and and the current solution is not optimal and ..., then for any slack variable, say* w_i*, we have* $w_i = 0$ *implies* $\Delta w_i \geq 0$*.*

- To paraphrase: for convex problems, as slack variables get small they tend to get large again. This is an antijamming theorem.
- A recent example of Wächter and Biegler shows that for nonconvex problems, jamming really can occur.
- Recent modification:
  - if a slack variable gets small and
  - its component of the step direction contributes to making a very short step,
  - then increase this slack variable to the average size of the variables the "mainstream" slack variables.
- This modification corrects all examples of jamming that we know about.

Nonconvex Optimization: Jamming

Theorem 3 *If the problem is convex and and the current solution is not optimal and ..., then for any slack variable, say* w_i*, we have* $w_i = 0$ *implies* $\Delta w_i \geq 0$*.*

- To paraphrase: for convex problems, as slack variables get small they tend to get large again. This is an antijamming theorem.
- A recent example of Wächter and Biegler shows that for nonconvex problems, jamming really can occur.
- Recent modification:
  - if a slack variable gets small and
  - its component of the step direction contributes to making a very short step,
  - then increase this slack variable to the average size of the variables the "mainstream" slack variables.
- This modification corrects all examples of jamming that we know about.

12 Nonconvex Optimization: Jamming

Theorem 3 *If the problem is convex and and the current solution is not optimal and ..., then for any slack variable, say* w_i*, we have* $w_i = 0$ *implies* $\Delta w_i \geq 0$*.*

- To paraphrase: for convex problems, as slack variables get small they tend to get large again. This is an antijamming theorem.
- A recent example of Wächter and Biegler shows that for nonconvex problems, jamming really can occur.
- Recent modification:
  - if a slack variable gets small and
  - its component of the step direction contributes to making a very short step,
  - then increase this slack variable to the average size of the variables the "mainstream" slack variables.
- This modification corrects all examples of jamming that we know about.

Nonconvex Optimization: Jamming

Theorem 3 *If the problem is convex and and the current solution is not optimal and ..., then for any slack variable, say* w_i*, we have* $w_i = 0$ *implies* $\Delta w_i \geq 0$*.*

- To paraphrase: for convex problems, as slack variables get small they tend to get large again. This is an antijamming theorem.
- A recent example of Wächter and Biegler shows that for nonconvex problems, jamming really can occur.
- Recent modification:
  - if a slack variable gets small and
  - its component of the step direction contributes to making a very short step,
  - then increase this slack variable to the average size of the variables the "mainstream" slack variables.
- This modification corrects all examples of jamming that we know about.

12 Nonconvex Optimization: Jamming

Theorem 3 *If the problem is convex and and the current solution is not optimal and ..., then for any slack variable, say* w_i*, we have* $w_i = 0$ *implies* $\Delta w_i \geq 0$*.*

- To paraphrase: for convex problems, as slack variables get small they tend to get large again. This is an antijamming theorem.
- A recent example of Wächter and Biegler shows that for nonconvex problems, jamming really can occur.
- Recent modification:
  - if a slack variable gets small and
  - its component of the step direction contributes to making a very short step,
  - then increase this slack variable to the average size of the variables the "mainstream" slack variables.
- This modification corrects all examples of jamming that we know about.

Nonconvex Optimization: Jamming

Theorem 3 *If the problem is convex and and the current solution is not optimal and ..., then for any slack variable, say* w_i*, we have* $w_i = 0$ *implies* $\Delta w_i \geq 0$*.*

- To paraphrase: for convex problems, as slack variables get small they tend to get large again. This is an antijamming theorem.
- A recent example of Wächter and Biegler shows that for nonconvex problems, jamming really can occur.
- Recent modification:
  - if a slack variable gets small and
  - its component of the step direction contributes to making a very short step,
  - then increase this slack variable to the average size of the variables the "mainstream" slack variables.
- This modification corrects all examples of jamming that we know about.

13 Modifications for General Problem Formulations

- Bounds, ranges, and free variables are all treated implicitly as described in Linear Programming: Foundations and Extensions (LP:F&E).
- Net result is following reduced KKT system:

$$\begin{bmatrix} -(H(x,y)+D) & A^T(x) \\ A(x) & E \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} \Phi_1 \\ \Phi_2 \end{bmatrix}$$

- Here, D and E are positive definite diagonal matrices.
- Note that D helps reduce frequency of diagonal perturbation.
- Choice of barrier parameter μ and initial solution, if none is provided, is described in the paper.
- Stopping rules, matrix reordering heuristics, etc. are as described in LP:F&E.

13 Modifications for General Problem Formulations

- Bounds, ranges, and free variables are all treated implicitly as described in Linear Programming: Foundations and Extensions (LP:F&E).
- Net result is following reduced KKT system:

$$\begin{bmatrix} -(H(x,y)+D) & A^T(x) \\ A(x) & E \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} \Phi_1 \\ \Phi_2 \end{bmatrix}$$

- Here, D and E are positive definite diagonal matrices.
- Note that D helps reduce frequency of diagonal perturbation.
- Choice of barrier parameter μ and initial solution, if none is provided, is described in the paper.
- Stopping rules, matrix reordering heuristics, etc. are as described in LP:F&E.

13 Modifications for General Problem Formulations

- Bounds, ranges, and free variables are all treated implicitly as described in Linear Programming: Foundations and Extensions (LP:F&E).
- Net result is following reduced KKT system:

$$\begin{bmatrix} -(H(x,y)+D) & A^T(x) \\ A(x) & E \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} \Phi_1 \\ \Phi_2 \end{bmatrix}$$

- Here, D and E are positive definite diagonal matrices.
- Note that D helps reduce frequency of diagonal perturbation.
- Choice of barrier parameter μ and initial solution, if none is provided, is described in the paper.
- Stopping rules, matrix reordering heuristics, etc. are as described in LP:F&E.

13 Modifications for General Problem Formulations

- Bounds, ranges, and free variables are all treated implicitly as described in Linear Programming: Foundations and Extensions (LP:F&E).
- Net result is following reduced KKT system:

$$\begin{bmatrix} -(H(x,y)+D) & A^T(x) \\ A(x) & E \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} \Phi_1 \\ \Phi_2 \end{bmatrix}$$

- Here, D and E are positive definite diagonal matrices.
- Note that D helps reduce frequency of diagonal perturbation.
- Choice of barrier parameter μ and initial solution, if none is provided, is described in the paper.
- Stopping rules, matrix reordering heuristics, etc. are as described in LP:F&E.

13 Modifications for General Problem Formulations

- Bounds, ranges, and free variables are all treated implicitly as described in Linear Programming: Foundations and Extensions (LP:F&E).
- Net result is following reduced KKT system:

$$\begin{bmatrix} -(H(x,y)+D) & A^T(x) \\ A(x) & E \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} \Phi_1 \\ \Phi_2 \end{bmatrix}$$

- Here, D and E are positive definite diagonal matrices.
- Note that D helps reduce frequency of diagonal perturbation.
- Choice of barrier parameter μ and initial solution, if none is provided, is described in the paper.
- Stopping rules, matrix reordering heuristics, etc. are as described in LP:F&E.

13 Modifications for General Problem Formulations

- Bounds, ranges, and free variables are all treated implicitly as described in Linear Programming: Foundations and Extensions (LP:F&E).
- Net result is following reduced KKT system:

$$\begin{bmatrix} -(H(x,y)+D) & A^T(x) \\ A(x) & E \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} \Phi_1 \\ \Phi_2 \end{bmatrix}$$

- Here, D and E are positive definite diagonal matrices.
- Note that D helps reduce frequency of diagonal perturbation.
- Choice of barrier parameter μ and initial solution, if none is provided, is described in the paper.
- Stopping rules, matrix reordering heuristics, etc. are as described in LP:F&E.

13 Modifications for General Problem Formulations

- Bounds, ranges, and free variables are all treated implicitly as described in Linear Programming: Foundations and Extensions (LP:F&E).
- Net result is following reduced KKT system:

$$\begin{bmatrix} -(H(x,y)+D) & A^T(x) \\ A(x) & E \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} \Phi_1 \\ \Phi_2 \end{bmatrix}$$

- Here, D and E are positive definite diagonal matrices.
- Note that D helps reduce frequency of diagonal perturbation.
- Choice of barrier parameter μ and initial solution, if none is provided, is described in the paper.
- Stopping rules, matrix reordering heuristics, etc. are as described in LP:F&E.

14 Computational Results

Compared:

- LOQO version 5.03 (20000528)
- SNOPT version 5.3-2 (Sep 1999), driver 19981124
- NITRO last week's version

Input/output interface: AMPL with Hessian info (recent enhancement due to hard work of David Gay).

14 Computational Results

Compared:

- LOQO version 5.03 (20000528)
- SNOPT version 5.3-2 (Sep 1999), driver 19981124
- NITRO last week's version

Input/output interface: AMPL with Hessian info (recent enhancement due to hard work of David Gay).

14 Computational Results

Compared:

- LOQO version 5.03 (20000528)
- SNOPT version 5.3-2 (Sep 1999), driver 19981124
- NITRO last week's version

Input/output interface: AMPL with Hessian info (recent enhancement due to hard work of David Gay).

14 Computational Results

Compared:

- LOQO version 5.03 (20000528)
- SNOPT version 5.3-2 (Sep 1999), driver 19981124
- NITRO last week's version

Input/output interface: AMPL with Hessian info (recent enhancement due to hard work of David Gay).

The CUTE/Schittkowski Set—Efficiency and Robustness

Size	Probs		SNOPT	NITRO	LOQO	LOQO tuned
Small	557	Num solved	535	476	532	16
$n < 100$		Tot time	87.22	191.99	69.32	9.21
Medium	63	Num solved	53	52	51	8
$n < 1000$		Tot time	551.46	690.33	254.47	189.49
Large	76	Num solved	38	60	71	1
$n < 10000$		Tot time	16032.99	13786.36	7357.81	0.63
Very Large	65	Num solved	11	13	49	12
$10000 \leq n$		Tot time	246026.22	249.71	49986.83	2744.93

16 The CUTE/Schittkowski Set—The Largest Ones

Out of the 65 very large problems, we looked at those which both LOQO and SNOPT solved. There were 8 such problems. From this 8, we have the following total times:

Number	SNOPT	LOQO
8	226844.88	1127.01

We also looked at those which both LOQO and NITRO solved. There were 13 such problems. From this 13, we have the following total times:

Number	NITRO	LOQO
13	249.71	240.11

16 The CUTE/Schittkowski Set—The Largest Ones

Out of the 65 very large problems, we looked at those which both LOQO and SNOPT solved. There were 8 such problems. From this 8, we have the following total times:

Number	SNOPT	LOQO
8	226844.88	1127.01

We also looked at those which both LOQO and NITRO solved. There were 13 such problems. From this 13, we have the following total times:

Number	NITRO	LOQO
13	249.71	240.11

16 The CUTE/Schittkowski Set—The Largest Ones

Out of the 65 very large problems, we looked at those which both LOQO and SNOPT solved. There were 8 such problems. From this 8, we have the following total times:

Number	SNOPT	LOQO
8	226844.88	1127.01

We also looked at those which both LOQO and NITRO solved. There were 13 such problems. From this 13, we have the following total times:

Number	NITRO	LOQO
13	249.71	240.11

16 The CUTE/Schittkowski Set—The Largest Ones

Out of the 65 very large problems, we looked at those which both LOQO and SNOPT solved. There were 8 such problems. From this 8, we have the following total times:

Number	SNOPT	LOQO
8	226844.88	1127.01

We also looked at those which both LOQO and NITRO solved. There were 13 such problems. From this 13, we have the following total times:

Number	NITRO	LOQO
13	249.71	240.11

16 The CUTE/Schittkowski Set—The Largest Ones

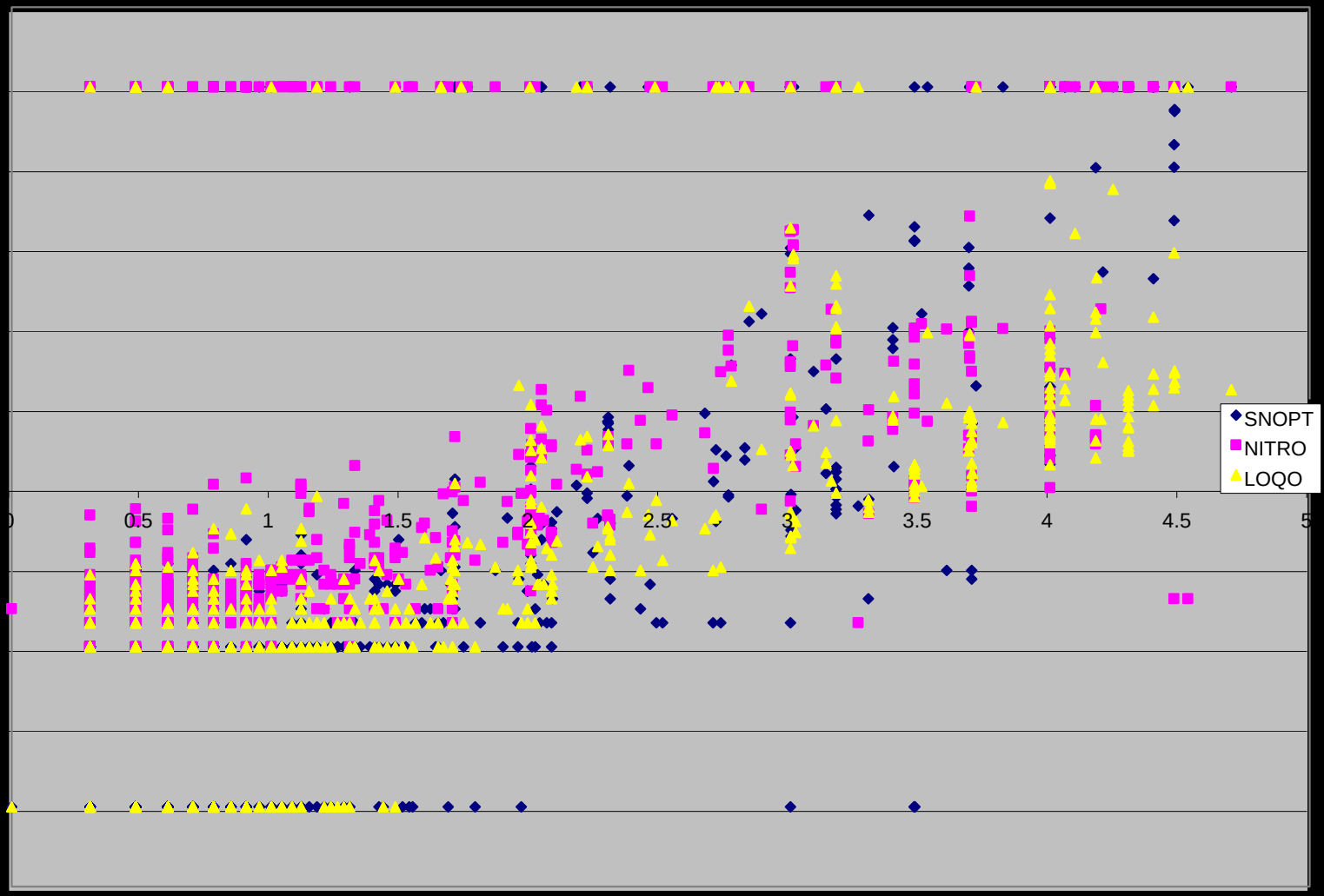
Out of the 65 very large problems, we looked at those which both LOQO and SNOPT solved. There were 8 such problems. From this 8, we have the following total times:

Number	SNOPT	LOQO
8	226844.88	1127.01

We also looked at those which both LOQO and NITRO solved. There were 13 such problems. From this 13, we have the following total times:

Number	NITRO	LOQO
13	249.71	240.11

17 Log-Log Plot of Time vs. Size



18 LOQO Failures

We have identified 5 issues that account for almost all of the LOQO failures:

- Failure to satisfy a constraint qualification. This means no KKT point.
- Unbounded set of optimal solutions. Interior-point methods go to the analytic center, which lies at infinity if the set is unbounded.
- Infeasible or unbounded problem. In nonconvex nonlinear programming, this is harder to detect than in LP.
- Nondifferentiability. If one of the nonlinear functions is not differentiable then one should anticipate trouble, especially if the nondifferentiability happens at the optimal solution.
- Bad Scaling. What can I say, shoot the modeler?

18 LOQO Failures

We have identified 5 issues that account for almost all of the LOQO failures:

- Failure to satisfy a constraint qualification. This means no KKT point.
- Unbounded set of optimal solutions. Interior-point methods go to the analytic center, which lies at infinity if the set is unbounded.
- Infeasible or unbounded problem. In nonconvex nonlinear programming, this is harder to detect than in LP.
- Nondifferentiability. If one of the nonlinear functions is not differentiable then one should anticipate trouble, especially if the nondifferentiability happens at the optimal solution.
- Bad Scaling. What can I say, shoot the modeler?

18 LOQO Failures

We have identified 5 issues that account for almost all of the LOQO failures:

- Failure to satisfy a constraint qualification. This means no KKT point.
- Unbounded set of optimal solutions. Interior-point methods go to the analytic center, which lies at infinity if the set is unbounded.
- Infeasible or unbounded problem. In nonconvex nonlinear programming, this is harder to detect than in LP.
- Nondifferentiability. If one of the nonlinear functions is not differentiable then one should anticipate trouble, especially if the nondifferentiability happens at the optimal solution.
- Bad Scaling. What can I say, shoot the modeler?

18 LOQO Failures

We have identified 5 issues that account for almost all of the LOQO failures:

- Failure to satisfy a constraint qualification. This means no KKT point.
- Unbounded set of optimal solutions. Interior-point methods go to the analytic center, which lies at infinity if the set is unbounded.
- Infeasible or unbounded problem. In nonconvex nonlinear programming, this is harder to detect than in LP.
- Nondifferentiability. If one of the nonlinear functions is not differentiable then one should anticipate trouble, especially if the nondifferentiability happens at the optimal solution.
- Bad Scaling. What can I say, shoot the modeler?

LOQO Failures

We have identified 5 issues that account for almost all of the LOQO failures:

- Failure to satisfy a constraint qualification. This means no KKT point.
- Unbounded set of optimal solutions. Interior-point methods go to the analytic center, which lies at infinity if the set is unbounded.
- Infeasible or unbounded problem. In nonconvex nonlinear programming, this is harder to detect than in LP.
- Nondifferentiability. If one of the nonlinear functions is not differentiable then one should anticipate trouble, especially if the nondifferentiability happens at the optimal solution.
- Bad Scaling. What can I say, shoot the modeler?

18 LOQO Failures

We have identified 5 issues that account for almost all of the LOQO failures:

- Failure to satisfy a constraint qualification. This means no KKT point.
- Unbounded set of optimal solutions. Interior-point methods go to the analytic center, which lies at infinity if the set is unbounded.
- Infeasible or unbounded problem. In nonconvex nonlinear programming, this is harder to detect than in LP.
- Nondifferentiability. If one of the nonlinear functions is not differentiable then one should anticipate trouble, especially if the nondifferentiability happens at the optimal solution.
- Bad Scaling. What can I say, shoot the modeler?

19 Nondifferentiability—s332.mod

In the Schittkowski model s332.mod, the following expressions appear:

$$\begin{aligned} y_i &= \arctan(\left|(1/t_i - x_1)/(\ln t_i + x_2)\right|), \\ p &= \max y_i, \\ p &\leq 30. \end{aligned}$$

Since $\arctan|x| = |\arctan x|$, the displayed expressions above are equivalent to

$$-30 \leq \arctan((1/t_i - x_1)/(\ln t_i + x_2)) \leq 30.$$

- This system is clearly differentiable, while the original system is not.
- All three codes failed on the problem as initially posed.
- But when restated, all solved the problem easily.
- SNOPT in .04 secs, NITRO in .53 secs, and LOQO in .1 secs.

19 Nondifferentiability—s332.mod

In the Schittkowski model s332.mod, the following expressions appear:

$$\begin{aligned} y_i &= \arctan(\left|(1/t_i - x_1)/(\ln t_i + x_2)\right|), \\ p &= \max y_i, \\ p &\leq 30. \end{aligned}$$

Since $\arctan|x| = |\arctan x|$, the displayed expressions above are equivalent to

$$-30 \leq \arctan((1/t_i - x_1)/(\ln t_i + x_2)) \leq 30.$$

- This system is clearly differentiable, while the original system is not.
- All three codes failed on the problem as initially posed.
- But when restated, all solved the problem easily.
- SNOPT in .04 secs, NITRO in .53 secs, and LOQO in .1 secs.

19 Nondifferentiability—s332.mod

In the Schittkowski model s332.mod, the following expressions appear:

$$\begin{aligned} y_i &= \arctan(\left|(1/t_i - x_1)/(\ln t_i + x_2)\right|), \\ p &= \max y_i, \\ p &\leq 30. \end{aligned}$$

Since $\arctan|x| = |\arctan x|$, the displayed expressions above are equivalent to

$$-30 \leq \arctan((1/t_i - x_1)/(\ln t_i + x_2)) \leq 30.$$

- This system is clearly differentiable, while the original system is not.
- All three codes failed on the problem as initially posed.
- But when restated, all solved the problem easily.
- SNOPT in .04 secs, NITRO in .53 secs, and LOQO in .1 secs.

19 Nondifferentiability—s332.mod

In the Schittkowski model s332.mod, the following expressions appear:

$$\begin{aligned} y_i &= \arctan(\left|(1/t_i - x_1)/(\ln t_i + x_2)\right|), \\ p &= \max y_i, \\ p &\leq 30. \end{aligned}$$

Since $\arctan |x| = |\arctan x|$, the displayed expressions above are equivalent to

$$-30 \leq \arctan((1/t_i - x_1)/(\ln t_i + x_2)) \leq 30.$$

- This system is clearly differentiable, while the original system is not.
- All three codes failed on the problem as initially posed.
- But when restated, all solved the problem easily.
- SNOPT in .04 secs, NITRO in .53 secs, and LOQO in .1 secs.

19 Nondifferentiability—s332.mod

In the Schittkowski model s332.mod, the following expressions appear:

$$\begin{aligned} y_i &= \arctan(\left|(1/t_i - x_1)/(\ln t_i + x_2)\right|), \\ p &= \max y_i, \\ p &\leq 30. \end{aligned}$$

Since $\arctan |x| = |\arctan x|$, the displayed expressions above are equivalent to

$$-30 \leq \arctan((1/t_i - x_1)/(\ln t_i + x_2)) \leq 30.$$

- This system is clearly differentiable, while the original system is not.
- All three codes failed on the problem as initially posed.
- But when restated, all solved the problem easily.
- SNOPT in .04 secs, NITRO in .53 secs, and LOQO in .1 secs.

19 Nondifferentiability—s332.mod

In the Schittkowski model s332.mod, the following expressions appear:

$$\begin{aligned} y_i &= \arctan(\left|(1/t_i - x_1)/(\ln t_i + x_2)\right|), \\ p &= \max y_i, \\ p &\leq 30. \end{aligned}$$

Since $\arctan |x| = |\arctan x|$, the displayed expressions above are equivalent to

$$-30 \leq \arctan((1/t_i - x_1)/(\ln t_i + x_2)) \leq 30.$$

- This system is clearly differentiable, while the original system is not.
- All three codes failed on the problem as initially posed.
- But when restated, all solved the problem easily.
- SNOPT in .04 secs, NITRO in .53 secs, and LOQO in .1 secs.

19 Nondifferentiability—s332.mod

In the Schittkowski model s332.mod, the following expressions appear:

$$\begin{aligned} y_i &= \arctan(\left|(1/t_i - x_1)/(\ln t_i + x_2)\right|), \\ p &= \max y_i, \\ p &\leq 30. \end{aligned}$$

Since $\arctan |x| = |\arctan x|$, the displayed expressions above are equivalent to

$$-30 \leq \arctan((1/t_i - x_1)/(\ln t_i + x_2)) \leq 30.$$

- This system is clearly differentiable, while the original system is not.
- All three codes failed on the problem as initially posed.
- But when restated, all solved the problem easily.
- SNOPT in .04 secs, NITRO in .53 secs, and LOQO in .1 secs.

20 Tuning

As we saw, LOQO solves most problems with the defaults but tuning did help in some cases. We only adjusted 5 tuning parameters. They are:

- bndpush. Most problems specify initial values for the variables. But, of course, they don't tell us how to initialize the slack variables that we add. `bndpush` gives a lower bound on the initial values for these slacks.
- sigfig. By default, LOQO asks for 8 digits of agreement between the primal and dual objective functions. For some problems this is just too much to expect.
- inftol. By default, LOQO asks for relative infeasibilities to be less than `1.0e-6` before declaring a primal or dual solution feasible. Again, sometimes this is too much to ask for.
- iterlim. By default, LOQO stops after a max of 200 iterations. Sometimes letting it go longer is a good thing.
- convex. Some problems are known to be convex. Asserting that that is the case sometimes helps.

20 Tuning

As we saw, LOQO solves most problems with the defaults but tuning did help in some cases. We only adjusted 5 tuning parameters. They are:

- bndpush. Most problems specify initial values for the variables. But, of course, they don't tell us how to initialize the slack variables that we add. `bndpush` gives a lower bound on the initial values for these slacks.
- sigfig. By default, LOQO asks for 8 digits of agreement between the primal and dual objective functions. For some problems this is just too much to expect.
- inftol. By default, LOQO asks for relative infeasibilities to be less than `1.0e-6` before declaring a primal or dual solution feasible. Again, sometimes this is too much to ask for.
- iterlim. By default, LOQO stops after a max of 200 iterations. Sometimes letting it go longer is a good thing.
- convex. Some problems are known to be convex. Asserting that that is the case sometimes helps.

20 Tuning

As we saw, LOQO solves most problems with the defaults but tuning did help in some cases. We only adjusted 5 tuning parameters. They are:

- bndpush. Most problems specify initial values for the variables. But, of course, they don't tell us how to initialize the slack variables that we add. `bndpush` gives a lower bound on the initial values for these slacks.
- sigfig. By default, LOQO asks for 8 digits of agreement between the primal and dual objective functions. For some problems this is just too much to expect.
- inftol. By default, LOQO asks for relative infeasibilities to be less than `1.0e-6` before declaring a primal or dual solution feasible. Again, sometimes this is too much to ask for.
- iterlim. By default, LOQO stops after a max of 200 iterations. Sometimes letting it go longer is a good thing.
- convex. Some problems are known to be convex. Asserting that that is the case sometimes helps.

20 Tuning

As we saw, LOQO solves most problems with the defaults but tuning did help in some cases. We only adjusted 5 tuning parameters. They are:

- bndpush. Most problems specify initial values for the variables. But, of course, they don't tell us how to initialize the slack variables that we add. `bndpush` gives a lower bound on the initial values for these slacks.
- sigfig. By default, LOQO asks for 8 digits of agreement between the primal and dual objective functions. For some problems this is just too much to expect.
- inftol. By default, LOQO asks for relative infeasibilities to be less than `1.0e-6` before declaring a primal or dual solution feasible. Again, sometimes this is too much to ask for.
- iterlim. By default, LOQO stops after a max of 200 iterations. Sometimes letting it go longer is a good thing.
- convex. Some problems are known to be convex. Asserting that that is the case sometimes helps.

20 Tuning

As we saw, LOQO solves most problems with the defaults but tuning did help in some cases. We only adjusted 5 tuning parameters. They are:

- bndpush. Most problems specify initial values for the variables. But, of course, they don't tell us how to initialize the slack variables that we add. `bndpush` gives a lower bound on the initial values for these slacks.
- sigfig. By default, LOQO asks for 8 digits of agreement between the primal and dual objective functions. For some problems this is just too much to expect.
- inftol. By default, LOQO asks for relative infeasibilities to be less than `1.0e-6` before declaring a primal or dual solution feasible. Again, sometimes this is too much to ask for.
- iterlim. By default, LOQO stops after a max of 200 iterations. Sometimes letting it go longer is a good thing.
- convex. Some problems are known to be convex. Asserting that that is the case sometimes helps.

20 Tuning

As we saw, LOQO solves most problems with the defaults but tuning did help in some cases. We only adjusted 5 tuning parameters. They are:

- bndpush. Most problems specify initial values for the variables. But, of course, they don't tell us how to initialize the slack variables that we add. `bndpush` gives a lower bound on the initial values for these slacks.
- sigfig. By default, LOQO asks for 8 digits of agreement between the primal and dual objective functions. For some problems this is just too much to expect.
- inftol. By default, LOQO asks for relative infeasibilities to be less than `1.0e-6` before declaring a primal or dual solution feasible. Again, sometimes this is too much to ask for.
- iterlim. By default, LOQO stops after a max of 200 iterations. Sometimes letting it go longer is a good thing.
- convex. Some problems are known to be convex. Asserting that that is the case sometimes helps.