Issues in Non-Convex Optimization

Robert M. Freund

with assistance from Brian W. Anthony

April 22, 2004

1 Outline

- General Nonlinear Optimization Problem
- Optimality Conditions for NLP
- Sequential Quadratic Programming (SQP) Method
- LOQO: Combining Interior-Point Methods and SQP
- Practical Issues in Solving NLP Problems

2 General Nonlinear Optimization Problem

$$\begin{aligned} \text{NLP:} \quad \text{minimize}_x \quad & f(x) \\ \text{s.t.} \quad & g_i(x) = 0, \quad i \in \mathcal{E} \\ & g_i(x) \leq 0, \quad i \in \mathcal{I} \\ & x \in \Re^n, \end{aligned}$$

where $f(x) : \Re^n \mapsto \Re$, $g_i(x) : \Re^n \mapsto \Re, i \in \mathcal{E} \cup \mathcal{I}$, $\mathcal{E}$ denotes the indices of the equality constraints, and $\mathcal{I}$ denotes the indices of the inequality constraints.

2.1 General Comments

Non-convex optimization problems arise in just about every economic and scientific domain:

- radiation therapy
- engineering product design
- economics: Nash equilibria
- finance: options pricing

- industrial engineering: traffic equilibria, supply chain management
- many other domains as well

Non-convex optimization is hard. Since $x - x^2 = 0$ if and only if $x \in \{0, 1\}$, we can formulate binary integer optimization as the following nonlinear optimization instance:

$$\begin{aligned} \text{BIP:} \quad \text{minimize}_x \quad & c^T x \\ \text{s.t.} \quad & Ax \leq b \\ & x_j - x_j^2 = 0, \ j = 1, \ldots, n \\ & x \in \Re^n \end{aligned}$$

2.2 Useful Definitions

The *feasible region* $\mathcal{F}$ of NLP is the set

$$\mathcal{F} = \{x \mid g_i(x) = 0 \text{ for } i \in \mathcal{E}, g_i(x) \leq 0 \text{ for } i \in \mathcal{I}\}$$

We have the following definitions of local/global, strict/non-strict minima/maxima.

Definition 2.1 *$\bar{x} \in \mathcal{F}$ is a* local minimum *of NLP if there exists $\epsilon > 0$ such that $f(\bar{x}) \leq f(x)$ for all $x \in B(\bar{x}, \epsilon) \cap \mathcal{F}$.*

Definition 2.2 *$\bar{x} \in \mathcal{F}$ is a* global minimum *of NLP if $f(\bar{x}) \leq f(x)$ for all $x \in \mathcal{F}$.*

Definition 2.3 *$\bar{x} \in \mathcal{F}$ is a* strict local minimum *of NLP if there exists $\epsilon > 0$ such that $f(\bar{x}) < f(x)$ for all $x \in B(\bar{x}, \epsilon) \cap \mathcal{F}$, $x \neq \bar{x}$.*

Definition 2.4 *$\bar{x} \in \mathcal{F}$ is a* strict global minimum *of NLP if $f(\bar{x}) < f(x)$ for all $x \in \mathcal{F}$, $x \neq \bar{x}$.*

Definition 2.5 $\bar{x} \in \mathcal{F}$ *is a* local maximum *of NLP if there exists* $\epsilon > 0$ *such that* $f(\bar{x}) \geq f(x)$ *for all* $x \in B(\bar{x}, \epsilon) \cap \mathcal{F}$.

Definition 2.6 $\bar{x} \in \mathcal{F}$ *is a* global maximum *of NLP if* $f(\bar{x}) \geq f(x)$ *for all* $x \in \mathcal{F}$.

Definition 2.7 $\bar{x} \in \mathcal{F}$ *is a* strict local maximum *of NLP if there exists* $\epsilon > 0$ *such that* $f(\bar{x}) > f(x)$ *for all* $x \in B(\bar{x}, \epsilon) \cap \mathcal{F}$, $x \neq \bar{x}$.

Definition 2.8 $\bar{x} \in \mathcal{F}$ *is a* strict global maximum *of NLP if* $f(\bar{x}) > f(x)$ *for all* $x \in \mathcal{F}$, $x \neq \bar{x}$.

If x is feasible for NLP, we let $\mathcal{I}(x)$ denote the indices of the active inequality constraints, namely:

$$\mathcal{I}(x) := \{i \in \mathcal{I} \mid g_i(x) = 0\} \ .$$

3 Optimality Conditions for NLP

Theorem: Karush-Kuhn-Tucker Necessary Conditions. Suppose that $f(x)$ and $g_i(x), i \in \mathcal{E} \cup \mathcal{I}$, are all differentiable functions. Under mild additional conditions, if $\bar{x}$ is a local minimum of NLP, then there exists $\bar{y}$ for which

$$(i) \quad \nabla f(\bar{x}) + \sum_{i \in \mathcal{E}} \bar{y}_i \nabla g_i(\bar{x}) + \sum_{i \in \mathcal{I}} \bar{y}_i \nabla g_i(\bar{x}) = 0$$

$$(ii) \quad g_i(\bar{x}) = 0, \ i \in \mathcal{E}$$

$$(iii) \quad g_i(\bar{x}) \leq 0, \ i \in \mathcal{I}$$

$$(iv) \quad \bar{y}_i \geq 0, \ i \in \mathcal{I}$$

$$(v) \quad \bar{y}_i \cdot g_i(\bar{x}) = 0, \ i \in \mathcal{I} \ .$$

q.e.d.

In the absence of convexity, a KKT point can be a global minimum, a local minimum, a "saddlepoint", or even a local or global maximum.

In order to develop sufficient conditions for a KKT point to be a local minimum, we need to work with the Lagrangian function associated with NLP, namely:

$$L(x, y) := f(x) + \sum_{i \in \mathcal{E}} y_i g_i(x) + \sum_{i \in \mathcal{I}} y_i g_i(x) \ .$$

We write $\nabla^2 f(x)$ for the Hessian matrix of $f(x)$ and we write

$$\nabla^2_{x,x} L(x, y)$$

for the Hessian matrix of $L(x, y)$ with respect to the x variables, namely:

$$\nabla^2_{x,x} L(x, y) = \nabla^2 f(x) + \sum_{i \in \mathcal{E}} y_i \nabla^2 g_i(x) + \sum_{i \in \mathcal{I}} y_i \nabla^2 g_i(x) \ . \tag{1}$$

We also need to work with the "cone of tangents" $\mathcal{K}(x)$ of a feasible point x, defined as:

$$\mathcal{K}(x) := \{d \mid \nabla g_i(x)^T d = 0 \text{ for } i \in \mathcal{E}, \nabla g_i(x)^T d \leq 0 \text{ for } i \in \mathcal{I}(x)\} \ .$$

Theorem: Karush-Kuhn-Tucker Sufficient Conditions. Suppose that $f(x)$ and $g_i(x), i \in \mathcal{E} \cup \mathcal{I}$, are all twice-differentiable functions. Suppose that $(\bar{x}, \bar{y})$ satisfy the following conditions:

$$(i) \quad \nabla f(\bar{x}) + \sum_{i \in \mathcal{E}} \bar{y}_i \nabla g_i(\bar{x}) + \sum_{i \in \mathcal{I}} \bar{y}_i \nabla g_i(\bar{x}) = 0$$

$$(ii) \quad g_i(\bar{x}) = 0, \ i \in \mathcal{E}$$

$$(iii) \quad g_i(\bar{x}) \leq 0, \ i \in \mathcal{I}$$

$$(iv) \quad \bar{y}_i \geq 0, \ i \in \mathcal{I}$$

$$(v) \quad \bar{y}_i \cdot g_i(\bar{x}) = 0, \ i \in \mathcal{I}$$

$$(vi) \quad d^T \left[\nabla^2_{x,x} L(\bar{x}, \bar{y})\right] d > 0 \text{ for } d \in \mathcal{K}(\bar{x}), d \neq 0 \ .$$

Then $\bar{x}$ is strict local minimum of NLP.
q.e.d.

3.1 Algorithm Issues

- It is rare that an algorithm for NLP will compute a global minimum.
- It is more usual for an algorithm to try to compute a local minimum, or at least to try to compute a KKT point.
- Most algorithms will achieve these goals "in the limit", in the sense that they generate a sequence which would converge to such a point if allowed to compute an infinite number of iterations.

Here is a quick overview of various types of algorithms that you have learned about already:

- Gradient-type methods
  - steepest descent
  - subgradient method
  - Frank-Wolfe (conditional gradient) method
  - conjugate gradient and/or conjugate directions methods

  These methods have low computational requirements at each iteration (few computations and little memory per iteration), but the convergence rates of these methods are at best linear, sometimes not even linear.
- Newton-type methods

  These methods have higher computational requirements at each iteration (much more computations and more memory per iteration), but convergence rates of these methods are usually locally quadratic.
- Interior-Point Methods

  These methods are best suited for convex optimization, but perform remarkably well on non-convex optimization as well. They usually have local quadratic convergence rates.

4 Sequential Quadratic Programming (SQP) Method

We consider the general nonlinear optimization problem:

$$\begin{aligned} \text{NLP:} \quad \text{minimize}_x \quad & f(x) \\ \text{s.t.} \quad & g_i(x) = 0, \quad i \in \mathcal{E} \\ & g_i(x) \leq 0, \quad i \in \mathcal{I} \\ & x \in \Re^n. \end{aligned}$$

The KKT conditions for this problem are that there exists $(\bar{x}, \bar{y})$ for which the following hold:

$$(i) \quad \nabla f(\bar{x}) + \sum_{i \in \mathcal{E}} \bar{y}_i \nabla g_i(\bar{x}) + \sum_{i \in \mathcal{I}} \bar{y}_i \nabla g_i(\bar{x}) = 0$$

$$(ii) \quad g_i(\bar{x}) = 0, \ i \in \mathcal{E}$$

$$(iii) \quad g_i(\bar{x}) \leq 0, \ i \in \mathcal{I}$$

$$(iv) \quad \bar{y}_i \geq 0, \ i \in \mathcal{I}$$

$$(v) \quad \bar{y}_i \cdot g_i(\bar{x}) = 0, \ i \in \mathcal{I}$$

The Lagrangian function associated with this problem is:

$$L(x, y) := f(x) + \sum_{i \in \mathcal{E}} y_i g_i(x) + \sum_{i \in \mathcal{I}} y_i g_i(x) \ .$$

4.1 First Key Idea of SQP Method: Solution of a Quadratic Problem with Primal and Dual Information

Suppose we have current iterate values $\bar{x}$ of the primal variables x and current iterate values $\bar{y}$ of the multipliers y. We can build a quadratic programming programming instance as follows. First, we use the second-order (quadratic) Taylor approximation of the objective function $f(x)$ at $\bar{x}$:

$$f(\bar{x}+\Delta x) \approx f(\bar{x}) + \nabla f(\bar{x})^T(\Delta x) + \tfrac{1}{2}(\Delta x)^T \nabla^2 f(\bar{x})(\Delta x) \ . \tag{2}$$

Second, we can replace the nonlinear constraints by their local linear approximation, and we obtain:

$$\begin{array}{lll} \mathrm{QP}_{\bar{x}}: & \mathrm{minimize}_{\Delta x} & \nabla f(\bar{x})^T(\Delta x) + \tfrac{1}{2}(\Delta x)^T \nabla^2 f(\bar{x})(\Delta x) \\ & \text{s.t.} & g_i(\bar{x}) + \nabla g_i(\bar{x})^T(\Delta x) = 0, \ i \in \mathcal{E} \\ & & g_i(\bar{x}) + \nabla g_i(\bar{x})^T(\Delta x) \leq 0, \ i \in \mathcal{I} \\ & & \Delta x \in \Re^n. \end{array} \tag{3}$$

However, this problem makes no use of the current values $\bar{y}$ of the multipliers y. Instead of using the Hessian of the original objective function $\nabla^2 f(\bar{x})$ in forming the objective function of the QP, SQP methods use the Hessian of the Lagrangian function:

$$\nabla^2_{x,x} L(\bar{x},\bar{y}) = \nabla^2 f(\bar{x}) + \sum_{i\in\mathcal{E}} \bar{y}_i \nabla^2 g_i(\bar{x}) + \sum_{i\in\mathcal{I}} \bar{y}_i \nabla^2 g_i(\bar{x}) \ .$$

This yields the following slightly modified QP problem:

$$\begin{array}{lll} \mathrm{QP}_{\bar{x},\bar{y}}: & \mathrm{minimize}_{\Delta x} & \nabla f(\bar{x})^T(\Delta x) + \tfrac{1}{2}(\Delta x)^T \left[\nabla^2_{x,x} L(\bar{x},\bar{y})\right](\Delta x) \\ & \text{s.t.} & g_i(\bar{x}) + \nabla g_i(\bar{x})^T(\Delta x) = 0, \ i \in \mathcal{E} \\ & & g_i(\bar{x}) + \nabla g_i(\bar{x})^T(\Delta x) \leq 0, \ i \in \mathcal{I} \\ & & \Delta x \in \Re^n. \end{array} \tag{4}$$

This problem is an "ordinary" quadratic problem (quadratic objective function and linear constraints). We can solve this problem and obtain Δx

as the primal solution and $\tilde{y}$ as the dual multipliers on the constraints. We then set:

$$\Delta y := \tilde{y} - \bar{y} \ .$$

In this way we use $(\bar{x}, \bar{y})$ to create directions $(\Delta x, \Delta y)$.

4.2 Second Key Idea of SQP Method: Merit Function to Measure Progress

Our current iterate is $(\bar{x}, \bar{y})$ and we have computed the direction $(\Delta x, \Delta y)$ from the solution of the quadratic program $\mathrm{QP}_{\bar{x},\bar{y}}$. We compute the new iterate values

$$(\bar{x}, \bar{y}) \leftarrow (\bar{x}, \bar{y}) + \bar{\alpha}(\Delta x, \Delta y)$$

using a step-size $\bar{\alpha}$. The typical method for choosing the step-size is to use a "merit function" that rewards improving the value of the original objective function $f(x)$ and penalizes for the extent of infeasibility. The most common general merit function is of the form:

$$P(x) := f(x) + \sum_{i \in \mathcal{E}} \rho_i |g_i(x)| + \sum_{i \in \mathcal{I}} \rho_i \max\{g_i(x), 0\} \ ,$$

where the penalty parameters $\rho_i, i \in \mathcal{E} \cup \mathcal{I}$, are user-specified. Thus $\bar{\alpha}$ is computed using:

$$\bar{\alpha} := \arg \min_{\alpha \in [0,1]} P(\bar{x} + \alpha \Delta x) \ .$$

4.3 General SQP Framework

The general algorithmic framework for the SQP method is as follows:

> **Step 1: Current Iterate.** We have current iterate values $\bar{x}$ of the primal variables x and current iterate values $\bar{y}$ of the multipliers y.
>
> **Step 2: Construct Quadratic Problem.** Use $(\bar{x}, \bar{y})$ to construct the quadratic program $\mathrm{QP}_{\bar{x},\bar{y}}$ as in (4).
>
> **Step 3: Solve Quadratic Problem and Construct $(\Delta x, \Delta y)$.** Solve the quadratic problem $\mathrm{QP}_{\bar{x},\bar{y}}$ and obtain primal solution Δx and (dual) multipliers $\tilde{y}$. Set $\Delta y = \tilde{y} - \bar{y}$.

Step 4: Compute Step-size and Update. Compute

$$\bar{\alpha} := \arg \min_{\alpha \in [0,1]} P(\bar{x} + \alpha \Delta x) \ .$$

Update iterate values $(\bar{x}, \bar{y}) \leftarrow (\bar{x}, \bar{y}) + \bar{\alpha}(\Delta x, \Delta y)$. Go to Step 2.

In some versions of the SQP method, the matrix $\nabla^2_{x,x} L(\bar{x}, \bar{y})$ is not computed exactly at each iteration. Instead, we work with an approximation $B(\bar{x}, \bar{y})$ of $\nabla^2_{x,x} L(\bar{x}, \bar{y})$ at each iteration, and $B(\bar{x}, \bar{y})$ is "updated" from iteration to iteration by simple rules and formulas that do not require much extra computation.

4.4 Where Does $\mathrm{QP}_{\bar{x},\bar{y}}$ Come From?

Herein we attempt to give some more insight/understanding regarding the choice of the quadratic program $\mathrm{QP}_{\bar{x},\bar{y}}$ and the resulting direction $(\Delta x, \Delta y)$. We will present an alternative derivation of the direction $(\Delta x, \Delta y)$ that shows how it arises from considerations of Newton's method applied to a KKT system related to the original problem. It is best for starters to assume that our problem only has equality constraints:

$$\begin{array}{lll} \text{ENLP:} \quad \text{minimize}_x & f(x) & \\ \text{s.t.} & g_i(x) = 0, \quad i \in \mathcal{E} & \\ & x \in \Re^n. & \end{array} \tag{5}$$

Then $\mathrm{QP}_{\bar{x},\bar{y}}$ for this problem is exactly:

$$\begin{array}{lll} \mathrm{QP}_{\bar{x},\bar{y}}: \quad \text{minimize}_{\Delta x} & \nabla f(\bar{x})^T (\Delta x) + \frac{1}{2} (\Delta x)^T \nabla^2_{x,x} L(\bar{x}, \bar{y})(\Delta x) & \\ \text{s.t.} & g_i(\bar{x}) + \nabla g_i(\bar{x})^T (\Delta x) = 0, \ i \in \mathcal{E} & \\ & \Delta x \in \Re^n. & \end{array} \tag{6}$$

Let $\tilde{y}$ denote the multipliers on the equality constraints. Because this quadratic problem has no inequalities, we can use (1) to write the KKT conditions of this system as:

$$\begin{aligned}
&\nabla f(\bar{x}) + \left[\nabla^2 f(\bar{x}) + \sum_{i\in\mathcal{E}} \bar{y}_i \nabla^2 g_i(\bar{x})\right](\Delta x) + \sum_{i\in\mathcal{E}} \tilde{y}_i \nabla g_i(\bar{x}) = 0 \\
&g_i(\bar{x}) + \nabla g_i(\bar{x})^T(\Delta x) = 0,\ i \in \mathcal{E}\ ,
\end{aligned} \tag{7}$$

Now let $\Delta y = \tilde{y} - \bar{y}$ and substitute:

$$\begin{aligned}
&\nabla f(\bar{x}) + \left[\nabla^2 f(\bar{x}) + \sum_{i\in\mathcal{E}} \bar{y}_i \nabla^2 g_i(\bar{x})\right](\Delta x) + \sum_{i\in\mathcal{E}} (\bar{y}_i + \Delta y_i) \nabla g_i(\bar{x}) = 0 \\
&g_i(\bar{x}) + \nabla g_i(\bar{x})^T(\Delta x) = 0,\ i \in \mathcal{E}\ .
\end{aligned} \tag{8}$$

Then the solution $(\Delta x, \Delta y)$ of (8) is the SQP direction.

We now show how the system of equations (8) in $(\Delta x, \Delta y)$ arises from Newton's method applied to the KKT system of the original problem. The KKT conditions for problem ENLP are:

$$\begin{aligned}
\text{KKT}: \quad &\nabla f(x) + \sum_{i\in\mathcal{E}} y_i \nabla g_i(x) = 0 \\
&g_i(x) = 0,\ i \in \mathcal{E}\ .
\end{aligned} \tag{9}$$

Our current point is $(x, y) = (\bar{x}, \bar{y})$, and let us consider moving in the direction $(\Delta x, \Delta y)$ to a point $(\bar{x}+\Delta x, \bar{y}+\Delta y)$. We would like $(\bar{x}+\Delta x, \bar{y}+\Delta y)$ to satisfy the KKT conditions for the problem ENLP, namely:

$$\begin{aligned}
\text{KKT}: \quad &\nabla f(\bar{x} + \Delta x) + \sum_{i\in\mathcal{E}} (\bar{y}_i + \Delta y_i) \nabla g_i(\bar{x} + \Delta x) = 0 \\
&g_i(\bar{x} + \Delta x) = 0,\ i \in \mathcal{E}\ .
\end{aligned} \tag{10}$$

Replacing each of the nonlinear terms with their linear approximations yields:

$$\nabla f(\bar{x}) + \nabla^2 f(\bar{x})(\Delta x) + \sum_{i \in \mathcal{E}} (\bar{y}_i + \Delta y_i) \Big(\nabla g_i(\bar{x}) + \nabla^2 g_i(\bar{x})(\Delta x) \Big) = 0$$

$$g_i(\bar{x}) + \nabla g_i(\bar{x})^T (\Delta x) = 0, \ i \in \mathcal{E} \ . \tag{11}$$

Finally, we delete the second-order terms "$\Delta y_i \nabla^2 g_i(\bar{x})(\Delta x)$" and rearrange the layout to yield the following completely linear system in $(\Delta x, \Delta y)$:

$$\nabla f(\bar{x}) + \left[\nabla^2 f(\bar{x}) + \sum_{i \in \mathcal{E}} \bar{y}_i \nabla^2 g_i(\bar{x}) \right] \Delta x + \sum_{i \in \mathcal{E}} (\bar{y}_i + \Delta y_i) \nabla g_i(\bar{x}) = 0$$

$$g_i(\bar{x}) + \nabla g_i(\bar{x})^T (\Delta x) = 0, \ i \in \mathcal{E} \ . \tag{12}$$

This is the Newton direction system associated with the original KKT conditions (9) for ENLP at $(\bar{x}, \bar{y})$. Now notice that the equation system (12) is a rearranged version the system (8), and so the solution $(\Delta x, \Delta y)$ of (12) and (8) are the same. Here we have shown that the SQP direction $(\Delta x, \Delta y)$ is the Newton direction for the KKT system for ENLP at $(\bar{x}, \bar{y})$.

5 LOQO: Combining Interior-Point Methods and SQP

The software LOQO has been developed by Professors Robert Vanderbei and David Shanno primarily over the last ten years. LOQO combines ideas from SQP and interior-point methods. Herein we describe the more salient features of the LOQO algorithm methodology. For this section it will be convenient to restrict our discussion to NLP problems with inequality constraints only, namely:

$$\begin{aligned} \text{INLP:} \quad \text{minimize}_x \qquad & f(x) \\ \text{s.t.} \qquad & g_i(x) \leq 0, \quad i \in \mathcal{I} \\ & x \in \Re^n. \end{aligned} \tag{13}$$

This problem is then converted to equality form by explicitly adding slack variables s and a logarithmic barrier function:

$$\begin{aligned} \text{BNLP}_\mu: \quad \text{minimize}_x \quad & f(x) - \mu \sum_{i \in \mathcal{I}} \ln(s_i) \\ \text{s.t.} \qquad & g(x) + s = 0 \\ & x \in \Re^n, s > 0. \end{aligned} \tag{14}$$

where

$$g(x) := (g_1(x), \ldots, g_m(x)) \ .$$

We then form the Lagrangian for this problem by assigning (dual) multipliers y to the constraints:

$$L(x, s, y) = f(x) - \mu \sum_{i \in \mathcal{I}} \ln(s_i) + y^T (g(x) + s) \ .$$

The KKT conditions for the problem BNLP_μ are:

$$\begin{aligned} & \nabla f(x) + y^T \nabla g(x) = 0 \\ & -\mu S^{-1} e + y = 0 \\ & g(x) + s = 0 \ , \end{aligned} \tag{15}$$

which we re-write equivalently as:

$$\begin{aligned} & \nabla f(x) + y^T \nabla g(x) = 0 \\ & -\mu e + SYe = 0 \\ & g(x) + s = 0 \ , \end{aligned} \tag{16}$$

Given iterate values $(\bar{x}, \bar{s}, \bar{y})$, the Newton equations for the above KKT system are:

$$\begin{pmatrix} \nabla^2 f(\bar{x}) + \sum_{i \in \mathcal{I}} \bar{y}_i \nabla^2 g_i(\bar{x}) & 0 & (\nabla g(\bar{x}))^T \\ 0 & \bar{Y} & \bar{S} \\ \nabla g(\bar{x}) & I & 0 \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta s \\ \Delta y \end{pmatrix} = \begin{pmatrix} -\nabla f(\bar{x}) - \bar{y}^T \nabla g(\bar{x}) \\ \mu e - \bar{S}\bar{Y}e \\ -g(\bar{x}) - \bar{s} \end{pmatrix} , \tag{17}$$

which using (1) is the same as:

$$\begin{pmatrix} \nabla^2_{x,x} L(\bar{x}, \bar{y}) & 0 & (\nabla g(\bar{x}))^T \\ 0 & \bar{Y} & \bar{S} \\ \nabla g(\bar{x}) & I & 0 \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta s \\ \Delta y \end{pmatrix} = \begin{pmatrix} -\nabla f(\bar{x}) - \bar{y}^T \nabla g(\bar{x}) \\ \mu e - \bar{S}\bar{Y}e \\ -g(\bar{x}) - \bar{s} \end{pmatrix} . \tag{18}$$

This system is essentially the same as an SQP system with the addition of logarthmic barrier function. LOQO solves this system at each iteration to obtain the iterate direction $(\Delta x, \Delta s, \Delta y)$. LOQO then updates the iterates using a step-size $\bar{\alpha}$ by computing:

$$(\bar{x}, \bar{s}, \bar{y}) \leftarrow (\bar{x}, \bar{s}, \bar{y}) + \bar{\alpha}(\Delta x, \Delta s, \Delta y) .$$

Here $\bar{\alpha}$ is chosen to minimize the following merit function:

$$P_{\mu,\rho}(x, s) = f(x) - \mu \sum_{i \in \mathcal{I}} \ln(s_i) + \rho \|g(x) + s\|^2 .$$

A more complete description of LOQO's algorithm methodology can be found in the following article:

"An interior point algorithm for nonconvex nonlinear programming" by R. Vanderbei and D. Shanno, *Computational Optimization and Applications* 13 (1999), pp. 231-252.

6 Illustration of Practical Issues in Solving NLP Problems

6.1 Illustration I: A Simple Covering Problem

Consider the problem of determining the smallest disk that contains the m given points $c_1, \ldots, c_m$. The decision variables in the problem are the center

x and the radius R of the containing disk. The problem has the simple formulation:

$$\begin{array}{lll} \text{CDP1}: & \text{minimize}_{x,R} & R \\ & \text{s.t.} & \|x - c_i\| \leq R, \quad i = 1, \ldots, m \\ & & x \in \Re^n, R \in \Re \ , \end{array}$$

where in this case $n = 2$ is the dimension in which the problem arises. As written the problem is convex but non-differentiable. Why might this be a problem for a solver?

6.1.1 Reformulations

We explore several different reformulations of the simple covering problem CDP1. We use these reformulations to become familiar with syntax and construction in Ampl (and LOQO). We attempt to solve the non-differentiable formulation of the problem as a means to explore some of the tricks and art that is necessary when we attempt to tackle non-convex problems.

The original simple covering problem can be reformulated in several ways. For example, if we square the non-differentiable constraint function and recognize that minimizing R is the same as minimizing R^2, we obtain:

$$\begin{array}{lll} \text{CDP2}: & \text{minimize}_{x,R} & R \ \ (\text{or} \ \ R^2) \\ & \text{s.t.} & x^T x - 2x^T c_i + c_i^T c_i \leq R^2 \ , \quad i = 1, \ldots, m \\ & & R \geq 0 \\ & & x \in \Re^n, R \in \Re \ , \end{array}$$

If we let $\delta = R^2$, we obtain a reformulation of the problem with a linear objective function and a quadratic constraint function:

$$\begin{array}{lll} \text{CDP3}: & \text{minimize}_{x,\delta} & \delta \\ & \text{s.t.} & x^T x - 2x^T c_i + c_i^T c_i - \delta \leq 0 \quad i = 1, \ldots, m \\ & & x \in \Re^n, \delta \in \Re \ , \end{array}$$

If we let $\alpha = x^T x - \delta$ in CDP3, we obtain a reformulation of the problem with a quadratic objective and a linear constraints:

$$\begin{array}{lll} \text{CDP4}: & \text{minimize}_{x,\alpha} & x^T x - \alpha \\ & \text{s.t.} & \alpha - 2x^T c_i + c_i^T c_i \leq 0 \quad i = 1, \ldots, m \\ & & x \in \Re^n, \alpha \in \Re \ , \end{array}$$

6.1.2 Solving CDP

We now explore the solution of the various formulations of the Simple Covering Problem using Ampl with LOQO. We used several very contrived data sets in order to explore and understand the behavior of the solution methods, and then we solved the problem using more realistic data sets. The data sets we used are listed in Table 1 and are shown in Figures 1, 2, 3, 4, and 5.

Data Set	Description
Cover1.dat	10×10 grid of points
Cover2.dat	random distribution of set of points
Cover3.dat	10×9 grid with 10 outliers
Cover4.dat	roughly 10×9 grid with 10 points near center
Cover5.dat	10×9 grid with 10 points near center

Table 1: Data sets for the Simple Covering Problem.

- The Ampl file **CDPlqnc.mod** is the original formulation (CDP1) of the packing problem. Try running this problem in Ampl with the data file *cover1.dat*. You should find that it does not solve. Why?
- The Ampl file **CDPql.mod** contains the formulation CDP4 (quadratic objective and linear constraints) version of the problem. **CDPql2.mod** contains the same formulation as **CDPql.mod** but demonstrates the *problem* command in Ampl. The Ampl file **CDPlq.mod** contains the formulation CDP3 (linear objective and quadratic constraints) version of the problem. All of these formulations of the problem solve very rapidly, with zero duality gap.
  - Experiment1: Solve **CDPql2.mod** with the LOQO convex option turned on and off. Is there a difference? If so, why?
  - Experiment2: Solve **CDPlq.mod** with the LOQO convex option turned on and off. Is there a difference? If so, why?
- The Ampl file **CDPlqnc2.mod** "solves" the original formulation CDP1. However, we first pre-solve the problem with one of the differentiable

formulations and then use this solution, modified slightly, as the starting point for the non-differentiable problem CDP1. The non-differentiable problem now solves. What does this suggest?

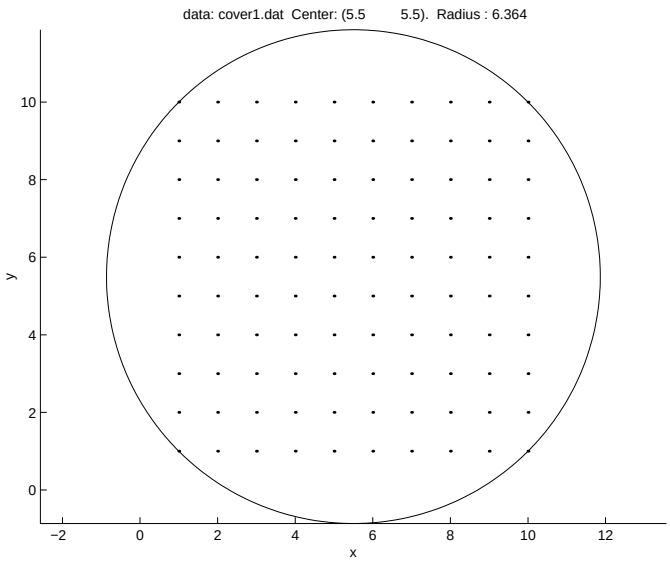

Figure 1: Data Set *Cover1.dat*

6.2 Illustration II: The Partial Covering Problem

Consider the following variation of the simple covering problem CDP, where we would like to find the smallest disk that contains 90% of the points. This problem is called the Partial Covering Problem. It arises in data mining, for example. The partial covering problem can be formulated as the following mixed integer problem:

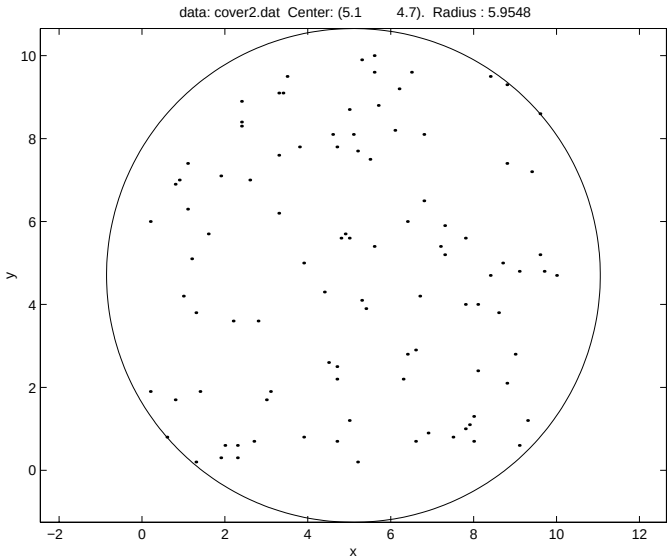

Figure 2: Data Set *Cover2.dat*

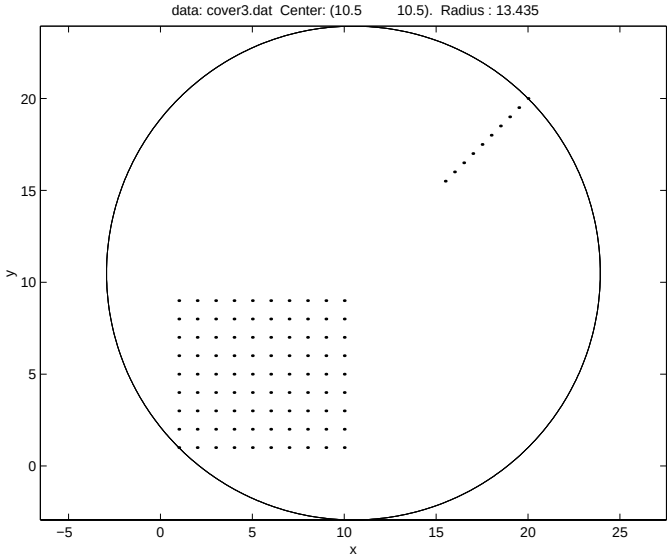

Figure 3: Data Set *Cover3.dat*

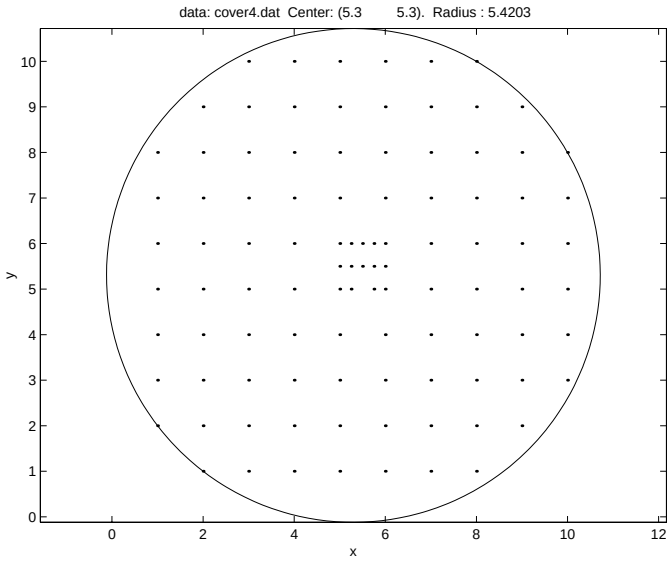

Figure 4: Data Set *Cover4.dat*

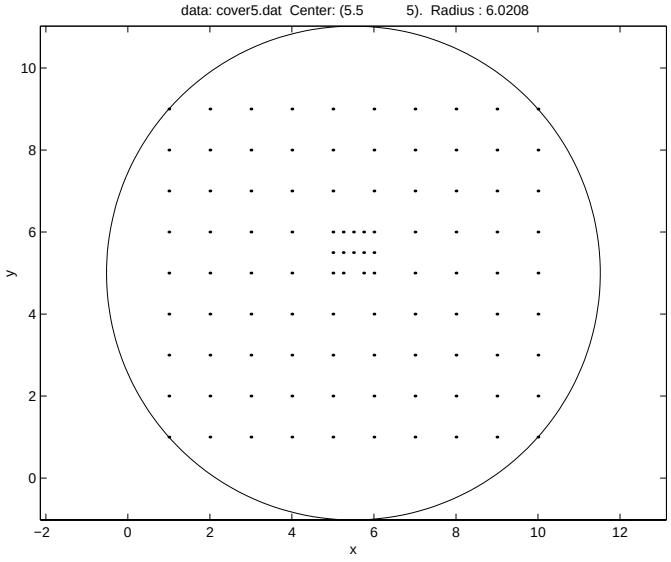

Figure 5: Data Set *Cover5.dat*

$$
\begin{array}{lll}
\text{IPCP}: \ \text{minimize}_{x,R,y,s} & R & \\
\text{s.t.} & \|x - c_i\| \leq R + s_i & i = 1, \ldots, m \\
& s_i \leq C y_i & i = 1, \ldots, m \\
& \sum_{i=1}^{m} y_i \leq 0.1m & \\
& x \in \Re^n, R \in \Re, s \geq 0, s \in \Re^m & \\
& y_i \in \{0, 1\}, i = 1, \ldots, m \ . &
\end{array}
$$

Here C is chosen beforehand as a large constant. In this formulation the nonnegative variable s_i is used to relax the constraint that forces c_i to be covered. The binary variable y_i is then turned on ($s_i \leq C y_i$) and the constraint $\sum_{i=1}^{m} y_i \leq 0.1m$ allows at most 10% of the points to be un-covered. This model should not be easy to solve. Why?

6.2.1 Reformulation of Partial Covering Problem using the Sigmoid Function

We use the sigmoid function:

$$f_\alpha(s) := \frac{1}{1 + e^{-\alpha s}}$$

to replace the integer variables y_i and the constraints "$s_i \leq C y_i$" in the model. The sigmoid function with parameter $\alpha > 0$ has the following attractive properties:

$f_\alpha(s) \to 0$ as $s \to -\infty$.

$f_\alpha(s) \to 1$ as $s \to +\infty$.

$1 - f_\alpha(s) = f_\alpha(-s)$.

$f_\alpha(0) = \frac{1}{2}$.

A graph of this function is shown in Figure 6. Note that if α is chosen sufficient large, then the shape of $f_\alpha(s)$ is almost the step function with step at $s = 0$.

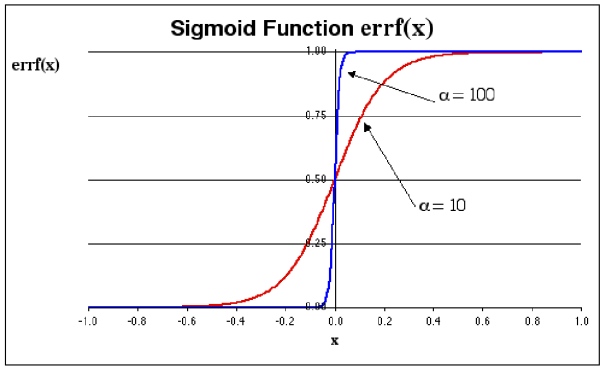

Figure 6: The sigmoid function.

Using the sigmoid function, our problem can be approximated as the following smooth nonlinear non-convex problem:

$$\begin{array}{lll} \text{NPCP}: \ \text{minimize}_{x,R,s} & R & \\ \text{s.t.} & \|x - c_i\| \leq R + s_i & i = 1, \ldots, m \\ & \sum_{i=1}^{m} f_\alpha(s_i) \leq 0.1m & \\ & x \in \Re^n, R \in \Re, s \in \Re^m \ . & \end{array}$$

6.2.2 Another Reformulation

Here we explore a reformulation of NPCP. (You are asked to develop other reformulations in the homework assignment.) If we square the non-differentiable constraint functions and recognize that minimizing R is the same as minimizing R^2, we obtain:

$$\begin{array}{lll} \text{NPCP2}: \ \text{minimize}_{x,R,s} & R \ \ (\text{or } R^2) & \\ \text{s.t.} & x^T x - 2x^T c_i + c_i^T c_i \leq R^2 + 2Rs_i + s_i^2 & i = 1, \ldots, m \\ & \sum_{i=1}^{m} f_\alpha(s_i) \leq 0.1m & \\ & R \geq 0 & \\ & x \in \Re^n, R \in \Re, s \in \Re^m \ . & \end{array}$$

The Ampl file **CDPlqsig1.mod** contains this formulation of the packing problem.

6.2.3 Solutions of the Partial Covering Problem

We solved the Partial Covering Problem using data sets *Cover1.dat* and *Cover3.dat*. Table 2 shows results of computational experiments with *Cover1.dat* for the 90% partial covering problem. Table 3 shows results of solving the 90% partial covering problem on the data set *Cover3.dat*. Table 4 shows results of solving the 89% partial covering problem on the data set *Cover3.dat*.

As Table 2 shows, we were able to select an appropriate value of α that worked rather well for *Cover1.dat*. Notice the sensitivity to the sigmoid smoothing parameter α. The larger that α becomes the more difficult the problem is to solve to a small duality gap. Why might this be so?

For the data set *Cover3.dat*, Table 3 shows clearly that there does not appear to be a reasonable combination of α and starting point (x_1^0, x_2^0) for which the algorithm works well. Notice that even if we initialize the solver at the known solution the problem $(x_1^0 = 5.5, x_2^0 = 5.0)$, the algorithm still moves to a non-global local optimum in all cases.

Table 4 shows computational results on data set *Cover3.dat* as the covering percentage is lowered from 90% to 89%. With this 1% relaxation we find that the value of the computed solution is mostly immune to initial conditions for small values of α (large values of α continue to make the problem hard to solve), and for $\alpha = 10$ the problem appears to be very insensitive to initial conditions.

	Initial Values				Solution Values					
α	x_1^0	x_2^0	R^0	Iterations	x_1^*	x_2^*	LOQO Primal Value	LOQO Dual Value	LOQO Duality Gap (%)	Actual Primal Gap (%)
5	5.5	5.5	6.634	114	5.50	5.50	5.60	5.60	0.00	3.27
6	5.5	5.5	6.634	79	5.50	5.50	5.58	5.58	0.00	3.03
6.5	5.5	5.5	6.634	1,000	5.40	5.43	5.58	5.59	0.11	2.99
7	5.5	5.5	6.634	1,000	5.41	5.35	5.58	5.60	0.39	2.98
10	5.5	5.5	6.634	1,000	5.31	5.33	5.59	5.68	1.76	3.04
20	5.5	5.5	6.634	1,000	5.31	5.32	5.59	5.67	1.34	3.14
50	5.5	5.5	6.634	1,000	5.60	5.59	5.64	5.91	4.71	4.13
100	5.5	5.5	6.634	1,000	5.39	5.47	5.63	5.10	-9.51	3.89
200	5.5	5.5	6.634	1,000	5.48	5.43	5.66	5.06	-10.59	4.40
Ideal Solution from *Cover4.dat* at 100%:					5.30	5.30	5.42			

Table 2: Solution of the 90% Partial Covering Problem Model CDP2 using LOQO for the data set *Cover1.dat*.

	Initial Values				Solution Values					
α	x_1^0	x_2^0	R^0	Iterations	x_1^*	x_2^*	LOQO Primal Value	LOQO Dual Value	LOQO Duality Gap (%)	Actual Primal Gap (%)
7	5.5	5.5	6.364	1000.00	5.50	5.00	8.34	8.48	1.68	38.50
7	5	5.5	6.364	1000.00	5.50	5.00	8.33	8.47	1.69	38.31
7	5.5	5	6.364	1000.00	5.33	4.77	10.87	>> 0	-	80.61
7	0	0	6.364	78.00	11.36	11.36	12.52	12.52	0.00	107.87
7	10.5	10.5	13.435	1000.00	5.50	5.00	8.32	8.47	1.69	38.24
7	5.5	5.5	0	1000.00	11.31	11.21	12.86	10.54	-22.00	113.58
7	5.5	5	6.0208	1000.00	5.66	5.18	10.28	>> 0	-	70.68
7	1	1	0	1000.00	5.50	5.00	8.36	8.50	1.68	38.80
10	5.5	5.5	6.364	1000.00	5.50	5.00	8.63	8.73	1.15	43.33
10	5	5.5	6.364	1000.00	5.50	5.00	8.40	8.50	1.18	39.52
10	5.5	5	6.364	1000.00	9.87	9.79	13.61	>> 0	-	126.06
10	0	0	6.364	1000.00	5.46	5.04	9.04	5.61	-61.25	50.22
10	10.5	10.5	13.435	1000.00	5.50	5.00	7.67	7.77	1.29	27.39
10	5.5	5.5	0	1000.00	5.50	5.00	7.67	7.77	1.29	27.35
10	5.5	5	6.0208	1000.00	5.38	4.96	7.82	>> 0	-	29.87
10	1	1	0	1000.00	5.70	5.29	15.91	>> 0	-	164.26
20	5.5	5.5	6.364	1000.00	9.89	10.26	12.58	>> 0	-	108.92
20	0	0	6.364	1000.00	-3.74	-2.78	28.17	>> 0	-	367.85
20	10.5	10.5	13.435	1000.00	11.87	12.43	11.98	>> 0	99.99	98.93
20	5.5	5	6.0208	1000.00	>> 0	<< 0	>> 0	>> 0	-	-
20	1	1	0	1000.00	5.40	4.88	6.51	>> 0	99.94	8.06
30	1	1	0	1000.00	-23.34	-21.76	36.23	>> 0	-	501.74
Ideal Solution from *Cover5.dat* at 100%:					5.50	5.00	6.0208			

Table 3: Solution of the 90% Partial Covering Problem Model CDP2 using LOQO for the data set *Cover3.dat*.

What might be a better reformulation of PCP? You will be asked to propose and solve an alternative formulation and in your homework assignment.

6.3 Illustration III: A Packing Problem

Consider the problem of packing a variety of wires of various radii into a cable. The m wires have given radii $r_1, r_2, \ldots, r_m$. We would like to determine the minimum width of a cable that will be used to enclose the wires. We can conceptualize this problem by considering a cross-section of the cable. The decision variables in the problem are the centers $x^1, \ldots, x^m$ of m disks of radii $r_1, \ldots, r_m$, and the radius R of the disk that is the cross-

	Initial Values				Solution Values					
α	x_1^0	x_2^0	R^0	Iterations	x_1^*	x_2^*	LOQO Primal Value	LOQO Dual Value	LOQO Duality Gap (%)	Actual Primal Gap (%)
7	10.5	10.5	13.435	52.00	5.50	5.00	6.18	6.18	0.00	2.69
10	10.5	10.5	13.435	69.00	5.50	5.00	6.13	6.13	0.00	1.83
15	10.5	10.5	13.435	109.00	5.50	5.00	6.09	6.09	0.00	1.22
20	10.5	10.5	13.435	84.00	11.42	11.42	12.29	12.29	0.00	104.08
30	10.5	10.5	13.435	1000.00	11.84	11.07	12.29	12.25	-0.34	104.14
7	5.5	5.5	6.634	48.00	5.50	5.00	6.18	6.18	0.00	2.69
10	5.5	5.5	6.634	146.00	5.50	5.00	6.13	6.13	0.00	1.83
15	5.5	5.5	6.634	1000.00	9.69	10.15	12.70	$<< 0$	-	110.95
20	5.5	5.5	6.634	1000.00	9.78	10.10	12.81	$<< 0$	-	112.74
7	1	1	0	128.00	5.50	5.00	6.18	6.18	0.00	2.69
10	1	1	0	94.00	5.50	5.00	6.13	6.13	0.00	1.83
15	1	1	0	1000.00	1.67	2.46	-0.01	$>> 0$	-	-100.24
7	0	0	0	177.00	5.50	5.00	6.18	6.18	0.00	2.69
10	0	0	0	183.00	5.50	5.00	6.13	6.13	0.00	1.83
15	0	0	0	121.00	11.43	11.43	12.32	12.32	0.00	104.65
Ideal Solution from *Cover5.dat* at 100%:					5.50	5.00	6.0208			

Table 4: Solution of the 89% Partial Covering Problem Model CDP2 using LOQO for the data set *Cover3.dat*.

section of the enclosing cable. This problem has the following formulation:

$$\begin{array}{lll} \text{PACK}: & \text{minimize}_{x^1,\ldots,x^m,R} & R \\ & \text{s.t.} & \|x^i - x^j\| \geq r_i + r_j \qquad 1 \leq i < j \leq m \\ & & \|x^i\| + r_i \leq R \qquad i = 1, \ldots, m \\ & & x^1, \ldots, x^m \in \Re^n, R \in \Re \ , \end{array}$$

where in this case $n = 2$ is the dimension in which the problem arises. The first set of constraints ensures that no two wires occupy the same space, and the second set of constraints ensures that the cable encloses all of the wires. As stated this problem is also a nonlinear non-convex optimization problem.

6.3.1 Reformulations of the Packing Problem

One problem with the constraints of PACK is that the constraint functions are not differentiable. We can reformulate PACK to alleviate this problem by squaring the constraints:

$$
\begin{array}{lll}
\text{PACK2}: \ \text{minimize}_{x^1,\ldots,x^m,R} & R \ (\text{or } R^2) & \\
\text{s.t.} & \|x^i - x^j\|^2 \geq (r_i + r_j)^2 & 1 \leq i < j \leq m \\
& \|x^i\|^2 \leq (R - r_i)^2 & i = 1, \ldots, m \\
& R - r_i \geq 0 & i = 1, \ldots, m \\
& x^1, \ldots, x^m \in \Re^n, R \in \Re \ , &
\end{array}
$$

Note that we add the constraints "$R - r_i \geq 0, i = 1, \ldots, m$. Why?

We could choose to square the objective (minimize R^2) or keep it as it was originally stated (minimize R). This may make a difference when attempting to solve the problem.

6.3.2 Solutions of the Packing Problem

The file **PP.mod** contains the Ampl code for the formulation PACK2 of the packing problem. As noted, this formulation is still nonlinear and non-convex. We expect to have difficulties solving this problem. We now know that we should start with a good initial guess of the solution. However, for the packing problem it is difficult to determine such an initial guess.

What are some potential algorithmic tricks or approaches that we might use to try to solve the packing problem?

- Start the optimization routine near an optimal solution. The issue here is how to construct the initial guess. You are invited to create and test your own heuristic methods for this approach.
- Try starting with a small subset of the disks, and add a new disk (and the associated new constraints) one at a time.
- Other ideas?

We started with two small disks as input and solved the problem. We then added another disk, and modified the previous solution in a way that we know will admit the new disk, and re-solved the resulting model again. In this way, at each iteration, we have a good initial condition and so we hope that we converge correctly to the global optimal solution of the original problem in this fashion. By adding disks one at a time, we are actually adding constraints (several at a time). If you want, you can think of this as

starting with a highly relaxed version of the problem and slowly remove the relaxation.

(The academic framework for the ideas above is the concept of *homotopy*. A homotopy is a continuous transformation from one problem to another. For example, a homotopy between a complicated function $f(x)$ and a simple function $g(x)$ from a space X to a space Y is a continuous map $G(x,t)$, from $X \times [0,1] \mapsto Y$ for which $G(x,0) = f(x)$ and $G(x,1) = g(x)$. We could try to solve for $f(x) = 0$ by starting at the easy-to-compute solution $\bar{x}$ of $g(x) = 0$ and tracing the path of solutions of $G(x,t) = 0$ starting with $G(\bar{x},1) = 0$ and shrinking t to 0.)

The file **PP2.com** contains the Ampl code for an iterative solution to the packing problem. The data files *Pack4.dat* and *Pack4a.dat* each contain lists of data for 50 disks, of which 25 disks have radius 5 and 25 disks have radius 10. In *Pack4.dat* the disks are ordered so that the 25 disks with radius 10 are listed first, followed by the 25 disks of radius 5. In *Pack4a.dat* the disks are ordered with alternating radii $10, 5, 10, 5, \ldots, 5$. Figures 7 and 8 show the final solutions for these two data sets. Notice that the different ordering of the disks have led to slightly different solutions with slightly different objective function values: 62.8256 for *Pack4.dat* versus 62.6141 for *Pack4a.dat*. We will show several movies of the iterations of the algorithm in class.

The data files *Pack43d.dat* and *Pack4a3d.dat* are data for a 3-dimensional version of the packing problem. Each data file contains lists of radii for 50 spheres, of which 25 spheres have radius 5 and 25 spheres have radius 10. In *Pack43d.dat* the spheres are ordered so that the 25 spheres with radius 10 are listed first, followed by the 25 spheres of radius 5. In *Pack4a3d.dat* the spheres are ordered with alternating radii $10, 5, 10, 5, \ldots, 5$. Figures 9 and 10 show the final solutions for these two data sets. Notice that the different ordering of the spheres have led to slightly different solutions with slightly different objective function values: 37.1878 for *Pack43d.dat* versus 37.6625 for *Pack4a3d.dat*. We will also show several movies of the iterations of the algorithm in class.

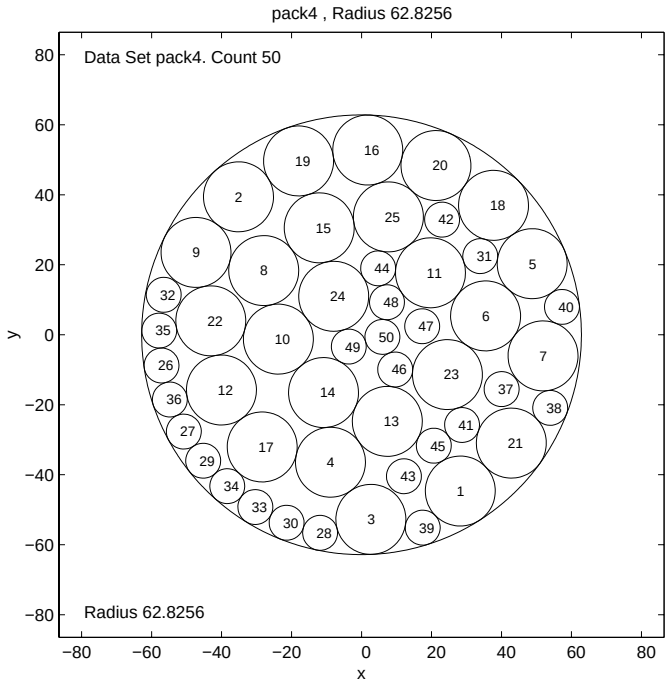

Figure 7: Packing Problem solution for data set *pack4.dat*.

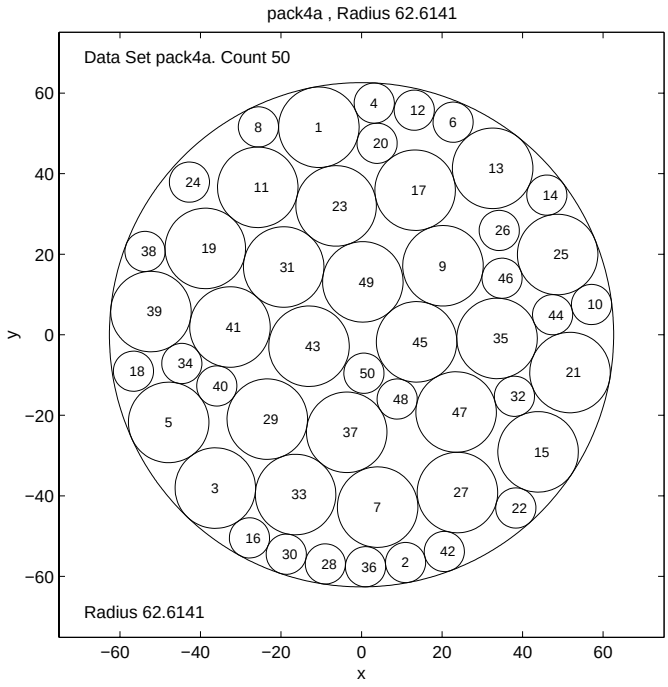

Figure 8: Packing Problem solution for data set *pack4a.dat*.

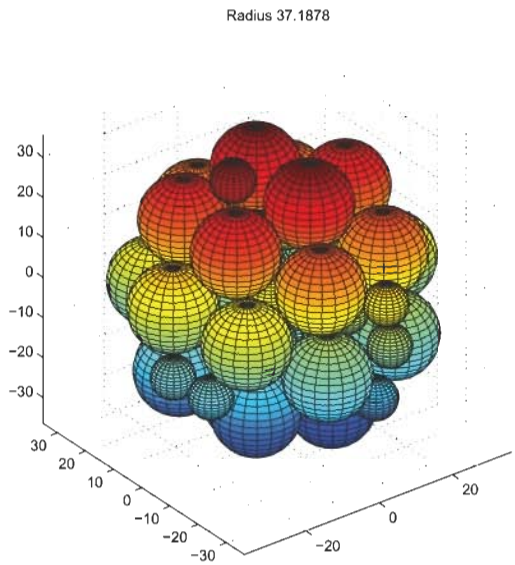

Figure 9: Packing Problem solution for data set *pack43d.dat*.

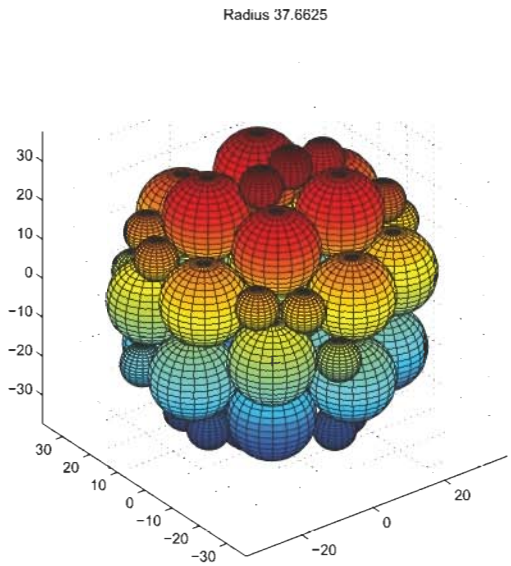

Figure 10: Packing Problem solution for data set *pack4a3d.dat*.