Analysis and Depiction of Solution Sets to Singular Nonlinear Constrained Optimization Problems

(Julie Roy's Ph.D. dissertation work, Spring, 2010)

Julie Roy
and
R. Baker Kearfott

Department of Mathematics
University of Louisiana at Lafayette

Algorithmic Differentiation, Optimization, and Beyond, April 8–9, 2010

Outline

Global Optimization

General Context and Notation

$$\begin{aligned} &\text{minimize } \varphi(x) \\ &\text{subject to } c_i(x) = 0,\ i = 1, \ldots, m_1, \\ &\qquad\qquad g_i(x) \leq 0,\ i = 1, \ldots, m_2, \end{aligned}$$

where the objective function $\varphi(x) : \boldsymbol{x} \to \mathbb{R}$ and the constraints $c_i, g_i : \boldsymbol{x} \to \mathbb{R}$ are possibly nonlinear, and where $\boldsymbol{x} \in \mathbb{R}^n$ is the box where $x_i \in [\underline{x}_i, \overline{x}_i]$ for $i = 1, \ldots, n$ defines the search region in a branch and bound algorithm.

- Our interest is in computing rigorous enclosures of all the optimizing points.

Singularities in Global Optimization

- For both linear and nonlinear global optimization problems, it often occurs that there are feasible (or approximately feasible) lines, planes, hyperplanes, or hypersurfaces that have approximately optimal objective function values.

Singularities in Global Optimization

- For both linear and nonlinear global optimization problems, it often occurs that there are feasible (or approximately feasible) lines, planes, hyperplanes, or hypersurfaces that have approximately optimal objective function values.

- There are two common problems encountered when solving singular global optimization problems using software with deterministic algorithms:

Singularities in Global Optimization

- For both linear and nonlinear global optimization problems, it often occurs that there are feasible (or approximately feasible) lines, planes, hyperplanes, or hypersurfaces that have approximately optimal objective function values.
- There are two common problems encountered when solving singular global optimization problems using software with deterministic algorithms:
  1. The algorithm only finds one optimal point and does not indicate that other solutions exist.

Singularities in Global Optimization

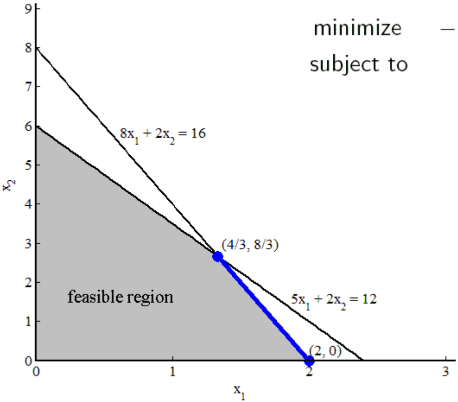
- For both linear and nonlinear global optimization problems, it often occurs that there are feasible (or approximately feasible) lines, planes, hyperplanes, or hypersurfaces that have approximately optimal objective function values.

- There are two common problems encountered when solving singular global optimization problems using software with deterministic algorithms:
    1. The algorithm only finds one optimal point and does not indicate that other solutions exist.
    2. The algorithm does not complete within a reasonable amount of time.

Depiction of Singular Solution Sets

Outline

Illustrative Example



$$\begin{array}{lrcl} \text{minimize} & -4x_1 - x_2 & & \\ \text{subject to} & 8x_1 + 2x_2 & \leq & 16, \\ & 5x_1 + 2x_2 & \leq & 12, \\ & x_1 & \geq & 0, \\ & x_2 & \geq & 0. \end{array}$$

The minimum value for the objective function over the feasible region is -8, and this value is attained at any point (x_1, x_2) on the line segment from $(4/3, 8/3)$ to $(2, 0)$.

Outline

Main Ideas

- In problems with singular (or approximately singular) solution sets, the singularity is first detected. Then, directions from an optimal point are found in which the objective function value remains approximately optimal and the points are still approximately feasible.

Main Ideas

- In problems with singular (or approximately singular) solution sets, the singularity is first detected. Then, directions from an optimal point are found in which the objective function value remains approximately optimal and the points are still approximately feasible.
- To visualize a singular solution set, one can construct a skewed box or boxes that contain points that are exact global optima and points that are within ϵ_φ of $\overline{\varphi}$ and within ϵ_c or ϵ_g of feasible.

Main Ideas

- In problems with singular (or approximately singular) solution sets, the singularity is first detected. Then, directions from an optimal point are found in which the objective function value remains approximately optimal and the points are still approximately feasible.
- To visualize a singular solution set, one can construct a skewed box or boxes that contain points that are exact global optima and points that are within ϵ_φ of $\overline{\varphi}$ and within ϵ_c or ϵ_g of feasible.
- These methods may be incorporated into a branch and bound process algorithm for global optimization to improve its efficiency.

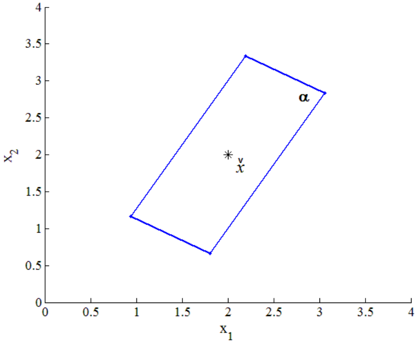
Depiction of Singular Solution Sets

Skewed Boxes

Definition

Let V be a square matrix whose columns are orthonormal vectors in $\mathbb{R}^n$ representing the axes in a coordinate system. A skewed box is an interval vector $\boldsymbol{\alpha} \in \mathbb{IR}^n$ centered at a point $\check{x} \in \mathbb{R}^n$ in the coordinate system defined by V.

Example



$$\boldsymbol{\alpha} = \begin{pmatrix} [-1.25,\ 1.25] \\ [-0.50,\ 0.50] \end{pmatrix}$$

$$\check{x} = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$$

$$V = \begin{pmatrix} \frac{1}{2} & -\frac{\sqrt{3}}{2} \\ \frac{\sqrt{3}}{2} & \frac{1}{2} \end{pmatrix}$$

Outline

Approximate Singular Solution Sets

Definition

Let $\check{x}$ be a feasible point such that $\varphi(\check{x})$ is approximately optimal. That is, $\varphi(x) \le \overline{\varphi} + \epsilon_\varphi$ for some tolerance ϵ_φ. If there are directions from $\check{x}$ in a coordinate system V in which the objective function $\varphi(x)$ remains approximately optimal and x remains approximately feasible, then an approximate singular solution set is a skewed box $\boldsymbol{\alpha}$ about $\check{x}$ such that for all $x \in \boldsymbol{\alpha}$:

1. $\varphi(x) \le \overline{\varphi} + \epsilon_\varphi$,
2. $|c_i(x)| \le \epsilon_c$ for $i = 1, \ldots, m_1$, and
3. $g_j(x) \le \epsilon_g$ for $j = 1, \ldots, m_2$.

Outline

Finding Directions Preserving Feasibility and Optimality

- Define the matrix G be by

$$G = \begin{pmatrix} \nabla^T \varphi(\check{x}) \\ \nabla^T c_1(\check{x}) \\ \vdots \\ \nabla^T c_{m_1}(\check{x}) \\ \nabla^T g_1(\check{x}) \\ \vdots \\ \nabla^T g_{n_a}(\check{x}) \\ \nabla^2 \varphi(\check{x}) \end{pmatrix}.$$

Finding Directions Preserving Feasibility and Optimality

- Define the matrix G be by

$$G = \begin{pmatrix} \nabla^T \varphi(\check{x}) \\ \nabla^T c_1(\check{x}) \\ \vdots \\ \nabla^T c_{m_1}(\check{x}) \\ \nabla^T g_1(\check{x}) \\ \vdots \\ \nabla^T g_{n_a}(\check{x}) \\ \nabla^2 \varphi(\check{x}) \end{pmatrix}.$$

- The null space of G contains directions in which points remain approximately optimal.

Finding Directions Preserving Feasibility and Optimality

- Define the matrix G be by

$$G = \begin{pmatrix} \nabla^T \varphi(\check{x}) \\ \nabla^T c_1(\check{x}) \\ \vdots \\ \nabla^T c_{m_1}(\check{x}) \\ \nabla^T g_1(\check{x}) \\ \vdots \\ \nabla^T g_{n_a}(\check{x}) \\ \nabla^2 \varphi(\check{x}) \end{pmatrix}.$$

- The null space of G contains directions in which points remain approximately optimal.
- The coordinate system consists of an orthonormal basis for the null space of G and its orthogonal complement.

Outline

Sufficient Conditions for a Skewed Box to be an Approximate Solution Box

- Initial approximate solution box coordinate widths, $|\alpha_i|$ ($i = 1, \ldots, n$), can be chosen such that

$$\sum_{i=1}^{n} |\alpha_i||D_{v_i}\varphi(\check{x})| + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} |\alpha_i||\alpha_j||D_{v_i,v_j}\varphi(\boldsymbol{x}^{(B)})| \leq \epsilon_\varphi,$$

$$\sum_{i=1}^{n} |\alpha_i||D_{v_i}c_k(\boldsymbol{x}^{(B)})| \leq \epsilon_c, \quad k = 1, \ldots, m_1, \text{ and}$$

$$\sum_{i=1}^{n} |\alpha_i||D_{v_i}g_\ell(\boldsymbol{x}^{(B)})| \leq \epsilon_g, \quad \ell = 1, \ldots, n_\text{a}.$$

where $\boldsymbol{x}^{(B)}$ is a small box whose center is $\check{x}$.

Sufficient Conditions for a Skewed Box to be an Approximate Solution Box

- Initial approximate solution box coordinate widths, $|\alpha_i|$ ($i = 1, \ldots, n$), can be chosen such that

$$\sum_{i=1}^{n} |\alpha_i||D_{v_i}\varphi(\check{x})| + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} |\alpha_i||\alpha_j||D_{v_i,v_j}\varphi(\boldsymbol{x}^{(B)})| \le \epsilon_\varphi,$$

$$\sum_{i=1}^{n} |\alpha_i||D_{v_i}c_k(\boldsymbol{x}^{(B)})| \le \epsilon_c, \quad k = 1, \ldots, m_1, \text{ and}$$

$$\sum_{i=1}^{n} |\alpha_i||D_{v_i}g_\ell(\boldsymbol{x}^{(B)})| \le \epsilon_g, \quad \ell = 1, \ldots, n_{\rm a}.$$

  where $\boldsymbol{x}^{(B)}$ is a small box whose center is $\check{x}$.
- In this heuristic, we evaluate the first and second order directional derivatives $D_{v_i,v_j}\varphi(\boldsymbol{x}^{(B)})$, $D_{v_i}c_k(\boldsymbol{x}^{(B)})$, and $D_{v_i}g_\ell(\boldsymbol{x}^{(B)})$ with interval arithmetic.

On Interval Evaluation of the Directional Derivatives

- If we evaluate the gradients and Hessian matrix with respect to the original coordinates first, and take dot products with the directions, this results in severe overestimation of the ranges of the directional derivatives.

On Interval Evaluation of the Directional Derivatives

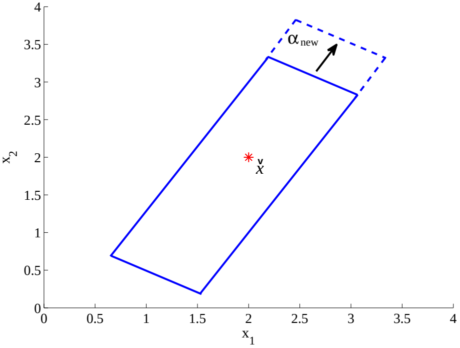
- If we evaluate the gradients and Hessian matrix with respect to the original coordinates first, and take dot products with the directions, this results in severe overestimation of the ranges of the directional derivatives.
- Instead, we compute the directional derivatives directly using automatic differentiation, we get usable bounds on their ranges.

On Interval Evaluation of the Directional Derivatives

- If we evaluate the gradients and Hessian matrix with respect to the original coordinates first, and take dot products with the directions, this results in severe overestimation of the ranges of the directional derivatives.
- Instead, we compute the directional derivatives directly using automatic differentiation, we get usable bounds on their ranges.
- Julie is presently polishing this second-order automatic differentiation code, written as MATLAB "`m`" files, for publication and release.

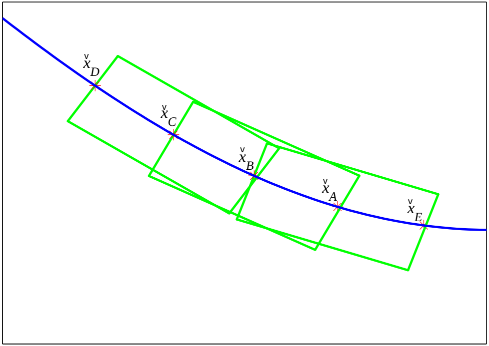
Expanding an Approximate Solution Box in the Optimal Directions

- Once an approximate solution box α is constructed, it is expanded outward from the sides of α in the directions in which the objective function values or constraint values are not changing much. This can be done using an epsilon-inflation type process.



Outline

Nonlinear Solution Sets

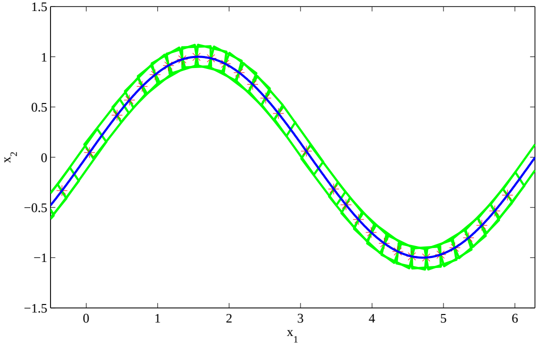


- For problems with nonlinear solution sets, a “chain” or “surface” of skewed boxes in different coordinate systems can be constructed to contain approximately optimal points.

Outline

Minimize $\varphi(x) = (\sin(x_1) - x_2)^2$

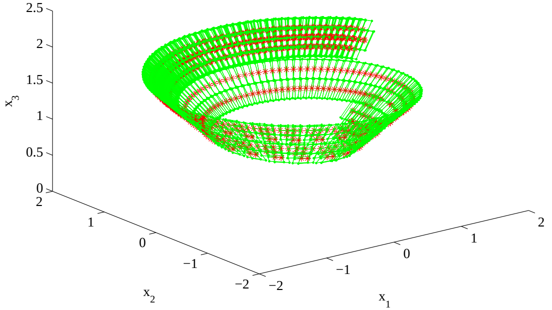
- The global minimum is 0, and all points on the curve $x_2 = \sin(x_1)$ are exact solutions. A portion of a chain of approximate solution boxes (computed with $\epsilon_\varphi = 0.1$ and $\check{x} \approx (6.283185, 0)^T$) is given in the figure.



blue curve - the exact solution set
red asterisks - approximately optimal points
green boxes - approximate solution boxes

Minimize $\varphi(x) = (x_1^2 + x_2^2 - x_3^2)^2$ Subject to $x_i \geq 0, i = 1, 2, 3$

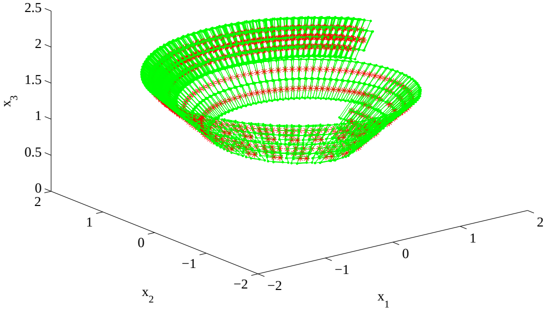
- The global minimum is 0, and the exact solution set for this problem is the lateral surface of a cone. A portion of a surface of approximate solution boxes is shown below.



red asterisks - approximately optimal points
green boxes - approximate solution boxes

Minimize $\varphi(x) = (x_1^2 + x_2^2 - x_3^2)^2$ Subject to $x_i \geq 0, i = 1, 2, 3$

- The global minimum is 0, and the exact solution set for this problem is the lateral surface of a cone. A portion of a surface of approximate solution boxes is shown below.



red asterisks - approximately optimal points
green boxes - approximate solution boxes

Incorporation into a Branch and Bound Algorithm

- When branch and bound algorithms are applied to singular problems, large number of small boxes with sides parallel to the coordinate axes are produced.

Incorporation into a Branch and Bound Algorithm

- When branch and bound algorithms are applied to singular problems, large number of small boxes with sides parallel to the coordinate axes are produced.
- In contrast, the singular solution set can be enclosed in a much smaller number of skewed boxes.

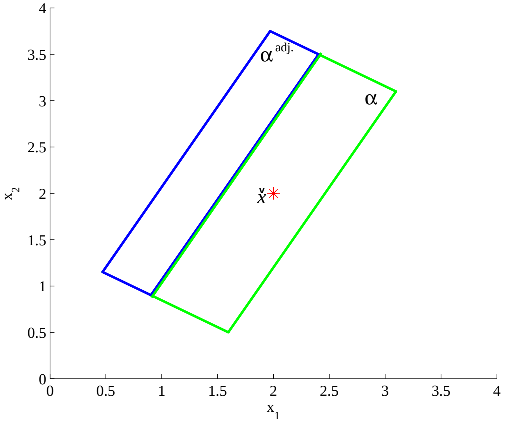
Incorporation into a Branch and Bound Algorithm

- When branch and bound algorithms are applied to singular problems, large number of small boxes with sides parallel to the coordinate axes are produced.
- In contrast, the singular solution set can be enclosed in a much smaller number of skewed boxes.
- We can construct a surface or chain of approximate solution boxes for a global optimization problem, then exclude all boxes within the branch and bound process that lie completely within a *rejection region*.

Outline

Rejection Regions

- The rejection region consists of approximate solution boxes and adjacent boxes. The adjacent boxes are constructed so that either at least one of the constraints is guaranteed to be infeasible or φ is guaranteed to be greater than $\overline{\varphi} + \epsilon$.



Rejection Regions

(Continued)

- More specifically, an adjacent box $\boldsymbol{\alpha}^{\text{adj.}}$ to a skewed box $\boldsymbol{\alpha}$ can be eliminated from the branch and bound search region if
  1. the objective function value for all of the points in $\boldsymbol{\alpha}^{\text{adj.}}$ is larger than $\overline{\varphi}$ (i.e., $\inf \varphi(\boldsymbol{\alpha}^{\text{adj.}}) > \overline{\varphi}$), or
  2. the points in $\boldsymbol{\alpha}^{\text{adj.}}$ are all exactly infeasible (i.e., $\text{mig}\, c_j(\boldsymbol{\alpha}^{\text{adj.}}) > 0$ for at least one of $j = 1, \ldots, m_1$ or $\inf g_j(\boldsymbol{\alpha}^{\text{adj.}}) > 0$ for at least one of $j = 1, \ldots, m_2$).

Rejection Regions

(Continued)

- More specifically, an adjacent box $\boldsymbol{\alpha}^{\text{adj.}}$ to a skewed box $\boldsymbol{\alpha}$ can be eliminated from the branch and bound search region if
  1. the objective function value for all of the points in $\boldsymbol{\alpha}^{\text{adj.}}$ is larger than $\overline{\varphi}$ (i.e., $\inf \varphi(\boldsymbol{\alpha}^{\text{adj.}}) > \overline{\varphi}$), or
  2. the points in $\boldsymbol{\alpha}^{\text{adj.}}$ are all exactly infeasible (i.e., $\text{mig}\, c_j(\boldsymbol{\alpha}^{\text{adj.}}) > 0$ for at least one of $j = 1, \ldots, m_1$ or $\inf g_j(\boldsymbol{\alpha}^{\text{adj.}}) > 0$ for at least one of $j = 1, \ldots, m_2$).
- The union of the adjacent skewed boxes with an approximate solution box $\boldsymbol{\alpha}$ is a large rejection region that can be eliminated from the search region in a branch and bound process.

Rejection Regions

(Continued)

- More specifically, an adjacent box $\boldsymbol{\alpha}^{\text{adj.}}$ to a skewed box $\boldsymbol{\alpha}$ can be eliminated from the branch and bound search region if
  1. the objective function value for all of the points in $\boldsymbol{\alpha}^{\text{adj.}}$ is larger than $\overline{\varphi}$ (i.e., $\inf \varphi(\boldsymbol{\alpha}^{\text{adj.}}) > \overline{\varphi}$), or
  2. the points in $\boldsymbol{\alpha}^{\text{adj.}}$ are all exactly infeasible (i.e., $\text{mig } c_j(\boldsymbol{\alpha}^{\text{adj.}}) > 0$ for at least one of $j = 1, \ldots, m_1$ or $\inf g_j(\boldsymbol{\alpha}^{\text{adj.}}) > 0$ for at least one of $j = 1, \ldots, m_2$).
- The union of the adjacent skewed boxes with an approximate solution box $\boldsymbol{\alpha}$ is a large rejection region that can be eliminated from the search region in a branch and bound process.
- Boxes constructed using the original coordinates in the branch and bound process can be rejected if they lie completely within the large rejection region.

Outline

Preliminary Results

- Julie programmed a simple branch and bound algorithm in MATLAB.

Preliminary Results

- Julie programmed a simple branch and bound algorithm in MATLAB.
- She tried the algorithm on several illustrative problems.

Preliminary Results

- Julie programmed a simple branch and bound algorithm in MATLAB.
- She tried the algorithm on several illustrative problems.
- On one, incorporation of the skewed rejection regions reduced the number of boxes considered from 15,565 to 4,414 and the time from 7.68 hours (interpretive, in MATLAB) to 43.58 minutes.

Preliminary Results

- Julie programmed a simple branch and bound algorithm in MATLAB.
- She tried the algorithm on several illustrative problems.
- On one, incorporation of the skewed rejection regions reduced the number of boxes considered from 15,565 to 4,414 and the time from 7.68 hours (interpretive, in MATLAB) to 43.58 minutes.
- On another, when `GlobSol` (a sophisticated compiled package) took many hours, inclusion of the rejection boxes took minutes in MATLAB.

Preliminary Results

- Julie programmed a simple branch and bound algorithm in MATLAB.
- She tried the algorithm on several illustrative problems.
- On one, incorporation of the skewed rejection regions reduced the number of boxes considered from 15,565 to 4,414 and the time from 7.68 hours (interpretive, in MATLAB) to 43.58 minutes.
- On another, when `GlobSol` (a sophisticated compiled package) took many hours, inclusion of the rejection boxes took minutes in MATLAB.
- We emphasize that this is preliminary work.

Julie's Dissertation is Available