Convexity Detection in Large-Scale Optimization

Robert Fourer

Industrial Engineering & Management Sciences
Northwestern University

AMPL Optimization

4er@northwestern.edu — 4er@ampl.com

OR53

Annual Conference of the OR Society

Nottingham — 6-8 September 2011

Convexity Detection in Large-Scale Optimization

Knowing that an optimization problem has a convex (or concave) objective and a convex constraint region is often valuable in computing a solution and interpreting the result. But determining *whether* a problem is convex is an intractable problem in general.

The first part of this presentation describes practical approaches to proving convexity, based on applying fundamental properties of convex functions to expressions built from standard mathematical functions. Among several possibilities, proofs of convexity may be constructed by recursively "walking" the expression graphs routinely produced by optimization modeling languages. Disproofs of convexity can also be valuable, but pose a quite different challenge involving a search for counterexamples.

The second part of the presentation turns to the unexpected complexities of detecting convex quadratic programs, which are amenable to efficient extensions of linear and mixed-integer programming solution methods. Although testing a quadratic function for convexity is easy, there is much more involved in testing whether a collection of quadratic inequalities defines a convex region. Particular interest has focused on conic regions and the "second-order cone programs" (or SOCPs) that they define. Whether given quadratic constraints define a convex cone can be determined numerically. But of equal interest are the numerous other objective and constraint types that have equivalent formulations as SOCPs. These include various combinations of sums and maxima of Euclidean norms, quadratic-linear ratios, products of powers, *p*-norms, and log-Chebychev terms. The tree-walk approach can be adapted to automatically detect and convert arbitrarily complex instances of these forms, freeing modelers from the time-consuming and error-prone work of maintaining the equivalent SOCPs explicitly.

Given an Optimization Model . . .

Does it have nice properties?

- Is it convex?
- Is it equivalent to a convex quadratic?

Does knowing that help to solve the problem?

- Are the results more believable?
- Are the computations more reliable?
- Are the computations more efficient?

Ways to Answer These Questions

Thought

- Theorems
- Equivalent formulations

Computation

- Detection algorithms
- Transformation algorithms
- Faster and more reliable
- Intractable in general
- Challenging in concept
- Challenging to implement

Introduction: Traffic Network

Given

N Set of nodes representing intersections

e Entrance to network

f Exit from network

$A \subseteq N \cup \{e\} \times N \cup \{f\}$

Set of arcs representing road links

and

b_{ij} Base travel time for each road link $(i, j) \in A$

c_{ij} Capacity for each road link $(i, j) \in A$

s_{ij} Traffic sensitivity for each road link $(i, j) \in A$

T Desired throughput from e to f

Example: Traffic Network

Determine

x_{ij} Traffic flow through road link $(i, j) \in A$

t_{ij} Actual travel time on road link $(i, j) \in A$

to minimize

$\Sigma_{(i,j)\in A}\, t_{ij} x_{ij} / T$

Average travel time from e to f

Example: Traffic Network

Subject to

$$t_{ij} = b_{ij} + \frac{s_{ij} x_{ij}}{1 - x_{ij}/c_{ij}} \quad \text{for all } (i,j) \in A$$

Travel times increase as flow approaches capacity

$$\Sigma_{(i,j)\in A}\, x_{ij} = \Sigma_{(j,i)\in A}\, x_{ji} \quad \text{for all } i \in N$$

Flow out equals flow in at any intersection

$$\Sigma_{(e,j)\in A}\, x_{ej} = T$$

Flow into the entrance equals the specified throughput

AMPL Traffic Network

Traffic network: symbolic data

```
set INTERS;           # intersections (network nodes)

param EN symbolic;    # entrance
param EX symbolic;    # exit

   check {EN,EX} not within INTERS;

set ROADS within {INTERS union {EN}} cross {INTERS union {EX}};

                         # road links (network arcs)

param base {ROADS} > 0;  # base travel times
param cap {ROADS} > 0;   # capacities
param sens {ROADS} > 0;  # traffic sensitivities

param through > 0;       # throughput
```

AMPL Traffic Network

Algebraic modeling language: symbolic model

```
var Flow {(i,j) in ROADS} >= 0, <= .9999 * cap[i,j];
var Time {ROADS} >= 0;

minimize Avg_Time:
  (sum {(i,j) in ROADS} Time[i,j] * Flow[i,j]) / through;

subject to Travel_Time {(i,j) in ROADS}:
  Time[i,j] = base[i,j] + (sens[i,j]*Flow[i,j]) / (1-Flow[i,j]/cap[i,j]);

subject to Balance_Node {i in INTERS}:
  sum{(i,j) in ROADS} Flow[i,j] = sum{(j,i) in ROADS} Flow[j,i];

subject to Balance_Enter:
  sum{(EN,j) in ROADS} Flow[EN,j] = through;
```

AMPL Traffic Network

Explicit data independent of symbolic model

```
set INTERS := b c ;

param EN := a ;
param EX := d ;

param: ROADS: base cap sens :=
         a b    4   10   .1
         a c    1   12   .7
         c b    2   20   .9
         b d    1   15   .5
         c d    6   10   .1 ;

param through := 20 ;
```

AMPL Traffic Network

Model + data = problem to solve, using KNITRO

```
ampl: model traffic.mod;
ampl: data traffic.dat;

ampl: option solver knitro;
ampl: solve;

KNITRO 7.0.0: Locally optimal solution.
objective 61.04695019; feasibility error 3.55e-14
12 iterations; 25 function evaluations

ampl: display Flow, Time;

:       Flow       Time    :=
a b     9.55146   25.2948
a c    10.4485    57.5709
b d    11.0044    21.6558
c b     1.45291    3.41006
c d     8.99562   14.9564
;
```

AMPL Traffic Network

Same with integer-valued variables

```
var Flow {(i,j) in ROADS} integer >= 0, <= .9999 * cap[i,j];
```

```
ampl: solve;

KNITRO 7.0.0: Locally optimal solution.
objective 76.26375; integrality gap 0
3 nodes; 5 subproblem solves

ampl: display Flow, Time;

:     Flow    Time   :=
a b     9     13
a c    11     93.4
b d    11     21.625
c b     2      4
c d     9     15
;
```

AMPL Traffic Network

Model + data = problem to solve, using CPLEX?

```
ampl: model traffic.mod;
ampl: data traffic.dat;

ampl: option solver cplex;
ampl: solve;

CPLEX 12.3.0.0:
Constraint _scon[1] is not convex quadratic
since it is an equality constraint.
```

AMPL Traffic Network

Look at the model again . . .

```
var Flow {(i,j) in ROADS} >= 0, <= .9999 * cap[i,j];
var Time {ROADS} >= 0;

minimize Avg_Time:
  (sum {(i,j) in ROADS} Time[i,j] * Flow[i,j]) / through;

subject to Travel_Time {(i,j) in ROADS}:
  Time[i,j] = base[i,j] + (sens[i,j]*Flow[i,j]) / (1-Flow[i,j]/cap[i,j]);

subject to Balance_Node {i in INTERS}:
  sum{(i,j) in ROADS} Flow[i,j] = sum{(j,i) in ROADS} Flow[j,i];

subject to Balance_Enter:
  sum{(EN,j) in ROADS} Flow[EN,j] = through;
```

AMPL Traffic Network

Quadratic reformulation

```
var Flow {(i,j) in ROADS} >= 0, <= .9999 * cap[i,j];
var Delay {ROADS} >= 0;

minimize Avg_Time:
  sum {(i,j) in ROADS} (base[i,j]*Flow[i,j] + Delay[i,j]) / through;

subject to Delay_Def {(i,j) in ROADS}:
  sens[i,j] * Flow[i,j]^2 <= (1 - Flow[i,j]/cap[i,j]) * Delay[i,j];

subject to Balance_Node {i in INTERS}:
  sum{(i,j) in ROADS} Flow[i,j] = sum{(j,i) in ROADS} Flow[j,i];

subject to Balance_Enter:
  sum{(EN,j) in ROADS} Flow[EN,j] = through;
```

AMPL Traffic Network

Model + data = problem to solve, using CPLEX?

```
ampl: model trafficQUAD.mod;
ampl: data traffic.dat;

ampl: option solver cplex;
ampl: solve;

CPLEX 12.3.0.0:
QP Hessian is not positive semi-definite.
```

AMPL Traffic Network

Quadratic reformulation #2

```
var Flow {(i,j) in ROADS} >= 0, <= .9999 * cap[i,j];
var Delay {ROADS} >= 0;
var Slack {ROADS} >= 0;

minimize Avg_Time:
  sum {(i,j) in ROADS} (base[i,j]*Flow[i,j] + Delay[i,j]) / through;

subject to Delay_Def {(i,j) in ROADS}:
  sens[i,j] * Flow[i,j]^2 <= Slack[i,j] * Delay[i,j];

subject to Slack_Def {(i,j) in ROADS}:
  Slack[i,j] = 1 - Flow[i,j]/cap[i,j];

subject to Balance_Node {i in INTERS}:
  sum{(i,j) in ROADS} Flow[i,j] = sum{(j,i) in ROADS} Flow[j,i];

subject to Balance_Enter:
  sum{(EN,j) in ROADS} Flow[EN,j] = through;
```

AMPL Traffic Network

Model + data = problem to solve, using CPLEX!

```
ampl: model trafficSOC.mod;
ampl: data traffic.dat;

ampl: option solver cplex;
ampl: solve;

CPLEX 12.3.0.0: primal optimal; objective 61.04693968
15 barrier iterations

ampl: display Flow;

Flow :=
a b    9.55175
a c   10.4482
b d   11.0044
c b    1.45264
c d    8.99561
;
```

AMPL Traffic Network

Same with integer-valued variables

```
var Flow {(i,j) in ROADS} integer >= 0, <= .9999 * cap[i,j];
```

```
ampl: solve;

CPLEX 12.3.0.0: optimal integer solution within mipgap or absmipgap;
   objective 76.26375017
19 MIP barrier iterations
0 branch-and-bound nodes

ampl: display Flow;

Flow :=
a b    9
a c   11
b d   11
c b    2
c d    9
;
```

Example: AC Power Network

Sines & cosines due to Kirchhoff's laws for AC

```
var G{(k,m) in YBUS} =

  if(k == m) then ...

else if(k != m) then
  sum {(l,k,m) in BRANCH}
    (- branch_g[l,k,m] * cos(branch_def[l,k,m])
     - branch_b[l,k,m] * sin(branch_def[l,k,m])) * branch_tap[l,k,m] +
  sum {(l,m,k) in BRANCH}
    (- branch_g[l,m,k] * cos(branch_def[l,m,k])
     + branch_b[l,m,k] * sin(branch_def[l,m,k])) * branch_tap[l,m,k];

minimize active_power :

  sum {k in BUS : bus_type[k] == 2 || bus_type[k] == 3}
   (bus_p_load[k]
    + sum {(k,m) in YBUS}
       bus_voltage[k] * bus_voltage[m]
         * (  G[k,m] * cos(bus_angle[k] - bus_angle[m])
            + B[k,m] * sin(bus_angle[k] - bus_angle[m])) )^2;
```

Outline

Recursive tree-walking algorithms

- Expression trees
- Detection algorithms
- Transformation algorithms

Convexity of general expressions

- Proof of convexity
- Disproof of convexity

Convexity of quadratic problems

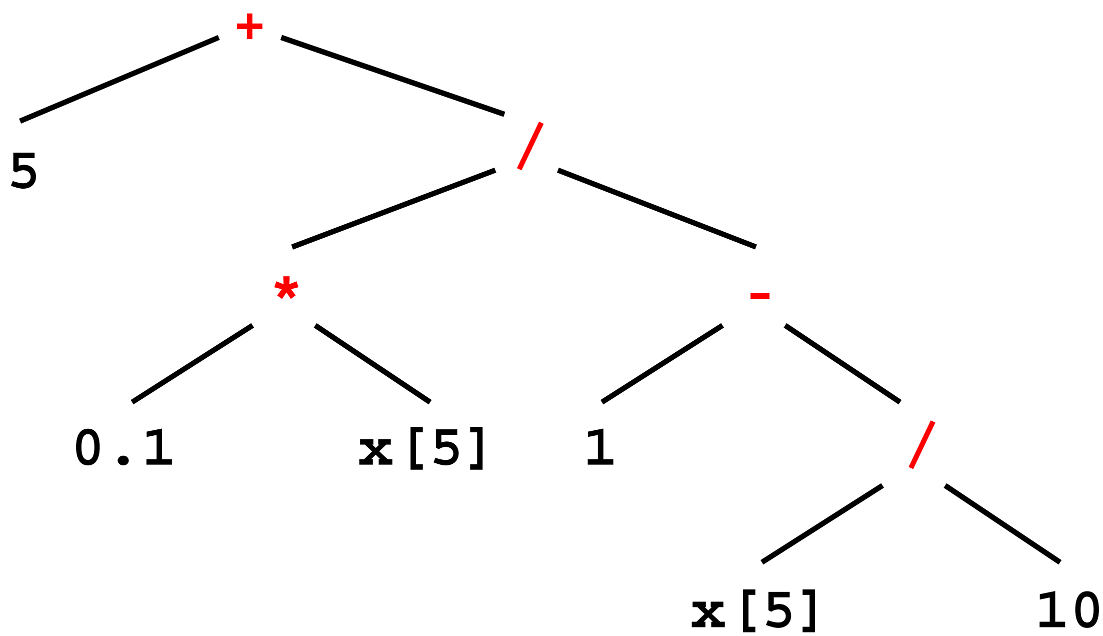
- Conic constraints
- Detection of equivalent problems
- Transformation of equivalent problems

Recursive Tree-Walking Algorithms

Expression

```
base[i,j] + (sens[i,j]*Flow[i,j]) / (1-Flow[i,j]/cap[i,j])
```

Expression tree



. . . actually a DAG

Detection: `isLinear()`

```
boolean isLinear (Node);

case of Node {

 PLUS:
 MINUS: return( isLinear(Node.left) and isLinear(Node.right) );

 TIMES: return( isConst(Node.left) and isLinear(Node.right) or
                isLinear(Node.left) and isConst(Node.right) );

 DIV:   return( isLinear(Node.left) and isConst(Node.right) );

 VAR:   return( TRUE );

 CONST: return( TRUE );

}
```

. . . to detect, test `isLinear(root)`

Detection: `isQuadr()`

```
boolean isQuadr (Node);

case of Node {

 PLUS:
 MINUS: return( isQuadr(Node.left) and isQuadr(Node.right) );

 TIMES: return( isLinear(Node.left) and isLinear(Node.right) or
                isQuadr(Node.left) and isConst(Node.right)   or
                isConst(Node.left) and isQuadr(Node.right) );

 POWER: return( isLinear(Node.left) and
                isConst(Node.right) and value(Node.right) == 2 );

 VAR:   return( TRUE );

 CONST: return( TRUE );

}
```

Transformation: `buildLinear()`

```
(coeff,const) = buildLinear (Node);

if Node.L then (coefL,consL) = buildLinear(Node.L);
if Node.R then (coefR,consR) = buildLinear(Node.R);

case of Node {

 PLUS:  coeff = mergeLists( coefL, coefR );
        const = consL + consR;

 TIMES: ...

 DIV:   coeff = coefL / consR;
        const = consL / consR;

 VAR:   coeff = makeList( 1, Node.index );
        const = 0;

 CONST: coeff = makeList( );
        const = Node.value;

}
```

. . . to transform, call `buildLinear(root)`

Convexity of General Expressions

Analyses

- Proof of convexity
- Disproof of convexity

Larger context ("DrAMPL")

- Classify problems based on AMPL output
- Recommend solvers

. . . joint project with Dominique Orban,
École Polytechnique de Montréal

Significance of Convexity

Theory

- For an optimization problem of the form

$$\boxed{\begin{array}{ll} \text{Minimize} & f(x_1,\ldots,x_n) \\ \text{Subject to} & g_i(x_1,\ldots,x_n) \ge 0, \quad i=1,\ldots,r \\ & h_i(x_1,\ldots,x_n) = 0, \quad i=1,\ldots,s \end{array}}$$

  a local minimum is global provided

  - ⋆ f is convex
  - ⋆ each g_i is convex
  - ⋆ each h_i is linear

Practice

- Many physical problems are naturally convex if formulated properly

Interest in Convexity Detection

Earlier approaches

- D.R. Stoutmeyer, "Automatic categorization of optimization problems: An application of computer symbolic mathematics." *Operations Research* **26** (1978) 773–788.
- I.P. Nenov, D.H. Fylstra and L.V. Kolev, "Convexity determination in the Microsoft Excel solver using automatic differentiation techniques." Fourth International Workshop on Automatic Differentiation (2004).
- M.C. Grant, S. Boyd and Y. Ye "Disciplined convex programming." In L. Liberti, N. Maculan, eds. *Global Optimization: From Theory to Implementation.* Springer, Nonconvex Optimization and Its Applications Series (2006) 155–210.

This work

- R. Fourer, C. Maheshwari, A. Neumaier, D. Orban and H. Schichl, "Convexity and Concavity Detection in Computational Graphs: Tree Walks for Convexity Assessment." dx.doi.org/10.1287/ijoc.1090.0321: *INFORMS Journal on Computing* **22** (2010) 26–43.

Proof of Convexity

Apply properties of functions

- $\|\mathbf{x}\|_p$ is convex, ≥ 0 everywhere
- x^α is convex for $\alpha \leq 0$, $\alpha \geq 1$; $-x^\alpha$ is convex for $0 \leq \alpha \leq 1$
- x^p for even $p > 0$ is convex everywhere,
  decreasing on $x \leq 0$, increasing on $x \geq 0$, *etc.*
- $-\log x$ and $x \log x$ are convex and increasing on $x > 0$
- $\sin x$ is concave on $0 \leq x \leq \pi$, convex on $\pi \leq x \leq 2\pi$,
  increasing on $0 \leq x \leq \pi/2$ and $3\pi/2 \leq x \leq 2\pi$, decreasing . . .
  ≥ -1 and ≤ 1 everywhere
- $e^{\alpha x}$ is convex, increasing everywhere for $\alpha > 0$, *etc.*
- $-(\Pi_i\, x_i)^{1/n}$ is convex where all $x_i > 0$

. . . ***etc., etc.***

Proof of Convexity *(cont'd)*

Apply properties of convexity

- Certain expressions are convex:
  - ⋆ $-f(\mathbf{x})$ for any concave f
  - ⋆ $\alpha f(\mathbf{x})$ for any convex f and $\alpha > 0$
  - ⋆ $f(\mathbf{x}) + g(\mathbf{x})$ for any convex f and g
  - ⋆ $f(\mathbf{Ax} + \mathbf{b})$ for any convex f
  - ⋆ $f(g(\mathbf{x}))$ for any convex nondecreasing f and convex g
  - ⋆ $f(g(\mathbf{x}))$ for any convex nonincreasing f and concave g
- Use these with function properties to assess convexity of node expressions on their domains

Apply properties of concavity, similarly

Proof of Convexity *(cont'd)*

Recursively apply `isConvex (lb,ub)`

- Return values
  - `+1: convex`
  - ` 0: can't tell`
  - `-1: concave`
- Bounds
  - `lb: lower bound`
  - `ub: upper bound`

Deduce status of each nonlinear expression

- Convex, concave, or indeterminate
- Lower and upper bounds

Disproof of Convexity

Find any counterexample

- Sample in feasible region
- Test any characterization of convex functions

Sampling along lines

- Look for $f(\lambda \mathbf{x}_1 + (1-\lambda)\mathbf{x}_2) > \lambda f(\mathbf{x}_1) + (1-\lambda) f(\mathbf{x}_2)$
- See implementation in John Chinneck's MProbe (www.sce.carleton.ca/faculty/chinneck/mprobe.html)

Sampling at points

- Look for $\nabla^2 f(\mathbf{x})$ not positive semi-definite
- Implemented in DrAMPL . . .

Disproof of Convexity *(cont'd)*

Sampling

- Choose points $\mathbf{x}_0$
  such that $x_{01}, \ldots, x_{0n}$ are within inferred bounds

Testing

- Apply GLTR (galahad.rl.ac.uk/galahad-www/doc/gltr.pdf) to

$$\begin{array}{ll} \min_{\mathbf{d}} & \nabla f(\mathbf{x}_0)\mathbf{d} + \tfrac{1}{2}\mathbf{d}\nabla^2 f(\mathbf{x}_0)\mathbf{d} \\ \text{s.t.} & \|\mathbf{d}\|_2 \le \max\{10, \|\nabla f(x_0)\|/10\} \end{array}$$

- Declare ***nonconvex*** if GLTR's Lanczos method
  finds a direction of negative curvature
- Declare ***inconclusive*** if GLTR
  reaches the trust region boundary
  without finding a direction of negative curvature

Testing Convexity Analyzers

Principles

- Disprovers can establish nonconvexity, suggest convexity
- Provers can establish convexity, suggest nonconvexity

Test problems

- Established test sets:
  COPS (17), CUTE (734), Hock & Schittkowski (119),
  Netlib (40), Schittkowski (195), Vanderbei (29 groups)
- Submissions to NEOS Server

Design of experiments

- Run a prover and a disprover on each test problem
- Check results for consistency
- Collect and characterize problems found to be convex
- Inspect functions not proved or disproved convex,
  to suggest possible enhancements to analyzers

Example

Torsion model (parameters and variables)

```
param nx > 0, integer;    # grid points in 1st direction
param ny > 0, integer;    # grid points in 2nd direction

param c;                  # constant

param hx := 1/(nx+1);     # grid spacing
param hy := 1/(ny+1);     # grid spacing

param area := 0.5*hx*hy;  # area of triangle

param D {i in 0..nx+1,j in 0..ny+1} =

  min( min(i,nx-i+1)*hx, min(j,ny-j+1)*hy );

                          # distance to the boundary

var v {i in 0..nx+1, j in 0..ny+1};

                          # definition of the
                          # finite element approximation
```

Example *(cont'd)*

Torsion model (objective and constraints)

```
var linLower = sum {i in 0..nx, j in 0..ny}
   (v[i+1,j] + v[i,j] + v[i,j+1]);

var linUpper = sum {i in 1..nx+1, j in 1..ny+1}
   (v[i,j] + v[i-1,j] + v[i,j-1]);

var quadLower = sum {i in 0..nx,j in 0..ny} (
   ((v[i+1,j] - v[i,j])/hx)**2 + ((v[i,j+1] - v[i,j])/hy)**2 );

var quadUpper = sum {i in 1..nx+1, j in 1..ny+1} (
   ((v[i,j] - v[i-1,j])/hx)**2 + ((v[i,j] - v[i,j-1])/hy)**2 );

minimize Stress:
  area * ((quadLower+quadUpper)/2 - c*(linLower+linUpper)/3);

subject to distanceBound {i in 0..nx+1, j in 0..ny+1}:
  -D[i,j] <= v[i,j] <= D[i,j];
```

Example *(cont'd)*

Output from AMPL's presolver

```
Presolve eliminates 2704 constraints and 204 variables.
Substitution eliminates 4 variables.

Adjusted problem:
2500 variables, all nonlinear
0 constraints
1 nonlinear objective; 2500 nonzeros.
```

Example *(cont'd)*

Output from DrAMPL (analysis)

```
Problem type
-------------
 -Problem has bounded variables
 -Problem has no constraints

Analyzing problem using only objective
---------------------------------------
 -This objective is quadratic
 -Problem is a QP with bounds

 -0.833013 <= objective <= 0.8359

Problem convexity
------------------
 Nonlinear objective looks convex on its domain.

 Detected 0/0 nonlinear convex  constraints,
          0/0 nonlinear concave constraints.
```

Issues

Algorithmic requirements

- Convexity outside feasible region

Nonconvex cases missed

- Choice of starting point can be crucial

Convex cases missed

- Polynomials
  - $x^4 - 4x^3 + 6x^2 - 4x + 1$ is $(x-1)^4$
  - $x^4 - 4x^3 + 7x^2 - 2x + 2$ is $(x-1)^4 + (x+1)^2$
- ***Quadratics . . .***

Convexity of Quadratic Expressions

“Elliptic” quadratic programming

- Detection
- Solving

“Conic” quadratic programming

- Detection
- Solving
- **Conversion**

. . . Ph.D. project of Jared Erickson,
Northwestern University

“Elliptic” Quadratic Programming

Symbolic detection

- Objectives
  - ∗ Minimize $x_1^2 + \ldots + x_n^2$
  - ∗ Minimize $\sum_{i=1}^{n} a_i(\mathbf{f}_i\mathbf{x} + g_i)^2,\ a_i \geq 0$
- Constraints
  - ∗ $x_1^2 + \ldots + x_n^2 \leq r$
  - ∗ $\sum_{i=1}^{n} a_i(\mathbf{f}_i\mathbf{x} + g_i)^2 \leq r,\ a_i \geq 0$

Numerical detection

- Objectives
  - ∗ Minimize $\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{q}\mathbf{x}$
- Constraints
  - ∗ $\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{q}\mathbf{x} \leq r$
- . . . where **Q** is positive semidefinite

Elliptic QP

Solving

Representation

- Much like LP
  - Coefficient lists for linear terms
  - Coefficient lists for quadratic terms
- No expression trees

Optimization

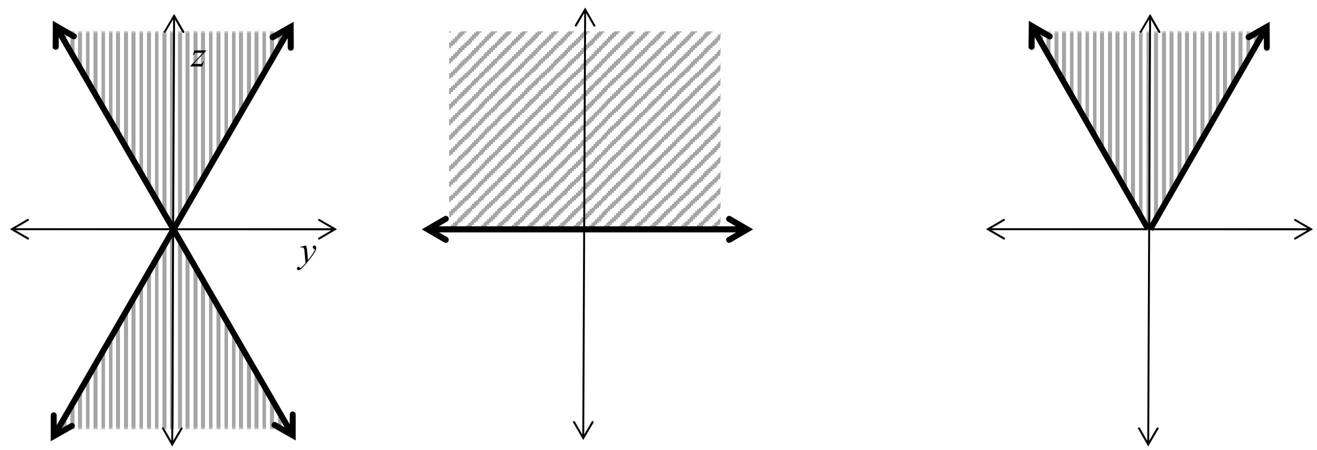
- Much like LP
  - Generalizations of barrier methods
  - Generalizations of simplex methods
  - Extensions of mixed-integer branch-and-bound schemes
- Simple derivative computations
- Less overhead than general-purpose nonlinear solvers

. . . ***your speedup may vary***

"Conic" Quadratic Programming

Standard cone

$x^2 + y^2 \leq z^2$ $\quad$ $z \geq 0$ $\quad$ $x^2 + y^2 \leq z^2,\ z \geq 0$



. . . boundary not smooth

Rotated cone

- $x^2 \leq yz,\ y \geq 0, z \geq 0, \ . \ . \ .$

"Conic" Quadratic Programming

Symbolic detection

- Constraints (standard)
  - $x_1^2 + \ldots + x_n^2 \leq x_{n+1}^2,\ \ x_{n+1} \geq 0$
  - $\sum_{i=1}^{n} a_i(\mathbf{f}_i\mathbf{x} + g_i)^2 \leq a_{n+1}(\mathbf{f}_{n+1}\mathbf{x} + g_{n+1})^2,$
    $a_1, \ldots, a_{n+1} \geq 0,\ \ \mathbf{f}_{n+1}\mathbf{x} + g_{n+1} \geq 0$
- Constraints (rotated)
  - $x_1^2 + \ldots + x_n^2 \leq x_{n+1}\, x_{n+2},\ x_{n+1} \geq 0,\ x_{n+2} \geq 0$
  - $\sum_{i=1}^{n} a_i(\mathbf{f}_i\mathbf{x} + g_i)^2 \leq a_{n+1}(\mathbf{f}_{n+1}\mathbf{x} + g_{n+1})(\mathbf{f}_{n+2}\mathbf{x} + g_{n+2}),$
    $a_1, \ldots, a_{n+1} \geq 0,\ \ \mathbf{f}_{n+1}\mathbf{x} + g_{n+1} \geq 0,\ \ \mathbf{f}_{n+2}\mathbf{x} + g_{n+2} \geq 0$

Numerical detection

- $\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{q}\mathbf{x} \leq r$
- . . . where $\mathbf{Q}$ has one negative eigenvalue
  - see Ashutosh Mahajan and Todd Munson, "Exploiting Second-Order Cone Structure for Global Optimization"

Conic QP

Solving

Similarities

- Describe by lists of coefficients
- Solve by extensions of LP barrier methods
- Extend to mixed-integer branch-and-bound

Differences

- Quadratic part not positive semi-definite
- Nonnegativity is essential
- Boundary of feasible region is not differentiable
- ***Many convex problems can be reduced to these . . .***

Terminology

- Second-order cone programs, SOCPs
- Allow also elliptical quadratic & linear constraints

SOCP-Solvable Forms

Quadratic

- Constraints (already seen)
- Objectives

SOC-representable

- Quadratic-linear ratios
- Generalized geometric means
- Generalized p-norms

Other objective functions

- Generalized product-of-powers
- Logarithmic Chebychev

SOCP-solvable

Quadratic

Standard cone constraints

- $\sum_{i=1}^{n} a_i(\mathbf{f}_i\mathbf{x} + g_i)^2 \le a_{n+1}(\mathbf{f}_{n+1}\mathbf{x} + g_{n+1})^2$,
  $a_1, \ldots, a_{n+1} \ge 0,\ \mathbf{f}_{n+1}\mathbf{x} + g_{n+1} \ge 0$

Rotated cone constraints

- $\sum_{i=1}^{n} a_i(\mathbf{f}_i\mathbf{x} + g_i)^2 \le a_{n+1}(\mathbf{f}_{n+1}\mathbf{x} + g_{n+1})(\mathbf{f}_{n+2}\mathbf{x} + g_{n+2})$,
  $a_1, \ldots, a_{n+1} \ge 0,\ \mathbf{f}_{n+1}\mathbf{x} + g_{n+1} \ge 0,\ \mathbf{f}_{n+2}\mathbf{x} + g_{n+2} \ge 0$

Sum-of-squares objectives

- Minimize $\sum_{i=1}^{n} a_i(\mathbf{f}_i\mathbf{x} + g_i)^2$
  - Minimize v
    Subject to $\sum_{i=1}^{n} a_i(\mathbf{f}_i\mathbf{x} + g_i)^2 \le v^2,\ v \ge 0$

SOCP-solvable

SOC-Representable

Definition

- Function $s(x)$ is SOC-representable *iff* . . .
- $s(x) \leq a_n(\mathbf{f}_{n+1}\mathbf{x} + g_{n+1})$ is equivalent to some combination of linear and quadratic cone constraints

Minimization property

- Minimize $s(x)$ is SOC-solvable
  - Minimize v_{n+1}
    Subject to $s(x) \leq v_{n+1}$

Combination properties

- $a \cdot s(x)$ is SOC-representable for any $a \geq 0$
- $\sum_{i=1}^{n} s_i(x)$ is SOC-representable
- $max_{i=1}^{n}\, s_i(x)$ is SOC-representable

. . . ***requires a recursive detection algorithm!***

SOCP-solvable

SOC-Representable (1)

Vector norm

- $\|\mathbf{a}\cdot(\mathbf{Fx}+\mathbf{g})\| = \sqrt{\sum_{i=1}^{n} a_i^2(\mathbf{f}_i\mathbf{x}+g_i)^2} \leq a_{n+1}(\mathbf{f}_{n+1}\mathbf{x}+g_{n+1})$
  - ∗ Square both sides to get standard SOC
    $\sum_{i=1}^{n} a_i^2(\mathbf{f}_i\mathbf{x}+g_i)^2 \leq a_{n+1}^2(\mathbf{f}_{n+1}\mathbf{x}+g_{n+1})^2$

Quadratic-linear ratio

- $\dfrac{\sum_{i=1}^{n} a_i(\mathbf{f}_i\mathbf{x}+g_i)^2}{\mathbf{f}_{n+2}\mathbf{x}+g_{n+2}} \leq a_{n+1}(\mathbf{f}_{n+1}\mathbf{x}+g_{n+1})$
  - ∗ Multiply by denominator to get rotated SOC
    $\sum_{i=1}^{n} a_i\,(\mathbf{f}_i\mathbf{x}+g_i)^2 \leq a_{n+1}\,(\mathbf{f}_{n+1}\mathbf{x}+g_{n+1})(\mathbf{f}_{n+2}\mathbf{x}+g_{n+2})$

SOCP-solvable

SOC-Representable (2)

Negative geometric mean

- $-\prod_{i=1}^{p}(\mathbf{f}_i\mathbf{x}+g_i)^{1/p} \le \mathbf{f}_{n+1}\mathbf{x}+g_{n+1},\ \ p \in \mathbb{Z}^+$
  - $-x_1^{1/4}x_2^{1/4}x_3^{1/4}x_4^{1/4} \le -x_5$ becomes rotated SOCs: $x_5^2 \le v_1v_2,\ v_1^2 \le x_1x_2,\ v_2^2 \le x_3x_4$
  - apply recursively $\lceil \log_2 p \rceil$ times

Generalizations

- $-\prod_{i=1}^{n}(\mathbf{f}_i\mathbf{x}+g_i)^{\alpha_i} \le a_{n+1}(\mathbf{f}_{n+1}\mathbf{x}+g_{n+1})$: $\sum_{i=1}^{n}\alpha_i \le 1,\ \alpha_i \in \mathbb{Q}^+$
- $\prod_{i=1}^{n}(\mathbf{f}_i\mathbf{x}+g_i)^{-\alpha_i} \le\ a_{n+1}(\mathbf{f}_{n+1}\mathbf{x}+g_{n+1}),\ \ \alpha_i \in \mathbb{Q}^+$

SOCP-solvable

SOC-Representable (3)

p-norm

- $\left(\sum_{i=1}^{n}|\mathbf{f}_i\mathbf{x}+g_i|^p\right)^{1/p} \le \mathbf{f}_{n+1}\mathbf{x}+g_{n+1},\ p \in \mathbb{Q}^+,\ p \ge 1$
  - $(|x_1|^5 + |x_2|^5)^{1/5} \le x_3$ can be written
    $|x_1|^5/x_3^4 + |x_2|^5/x_3^4 \le x_3$ which becomes
    $v_1 + v_2 \le x_3$ with $-v_1^{1/5}\, x_3^{4/5} \le \pm x_1, -v_1^{1/5}\, x_3^{4/5} \le \pm x_2$
  - reduces to product of powers

Generalizations

- $\left(\sum_{i=1}^{n}|\mathbf{f}_i\mathbf{x}+g_i|^{\alpha_i}\right)^{1/\alpha_0} \le \mathbf{f}_{n+1}\mathbf{x}+g_{n+1},\ \alpha_i \in \mathbb{Q}^+,\ \alpha_i \ge\ \alpha_0 \ge 1$
- $\sum_{i=1}^{n}|\mathbf{f}_i\mathbf{x}+g_i|^{\alpha_i} \le (\mathbf{f}_{n+1}\mathbf{x}+g_{n+1})^{\alpha_0}$
- Minimize $\sum_{i=1}^{n}|\mathbf{f}_i\mathbf{x}+g_i|^{\alpha_i}$

. . . *standard SOCP has $\boldsymbol{\alpha_i \equiv 2}$*

SOCP-solvable

Other Objective Forms

Unrestricted product of powers

- Minimize $-\prod_{i=1}^{n}(\mathbf{f}_i\mathbf{x}+g_i)^{\alpha_i}$ for any $\alpha_i \in \mathbb{Q}^+$

Logarithmic Chebychev approximation

- Minimize $\max_{i=1}^{n}|\log(\mathbf{f}_i\mathbf{x})-\log(g_i)|$

Why no constraint versions?

- Not SOC-representable
- Transformation changes objective value (but not solution)

Example: Sum-of-Norms Objective

Given

- Minimize $\sum_{i=1}^{m} a_i \sqrt{\sum_{j=1}^{n} \left(\mathbf{f}_{ij}\mathbf{x} + g_{ij}\right)^2 + c_i}$

Transform to

- Minimize $\sum_{i=1}^{m} s_i$
- $\sum_{j=1}^{n} t_{ij}^2 + a_i^2 c_i \le s_i^2,\ \ s_i \ge 0, i = 1, \ldots, m$
- $a_i\left(\mathbf{f}_{ij}\mathbf{x} + g_{ij}\right) = t_{ij}\,, \qquad\qquad j = 1, \ldots, n$

Sum of Norms

Detection

```
boolean isSumNorms (Node);
case of Node {
 PLUS:  return( isSumNorms(Node.left) and isSumNorms(Node.right) );
 TIMES: return( isSumNorms(Node.right) and
                isConst(Node.left) and value(Node.left) > 0 );
 SQRT:  return( isNormSquared(Node.child) );
}

boolean isSumSquares (Node);
case of Node {
 PLUS:  return( isSumSquares(Node.left) and isSumSquares(Node.right) );
 POWER: return( isLinear(Node.left) and
                isConst(Node.right) and value(Node.right) == 2 );
 CONST: return( value(Node) > 0 );
}
```

Transformation: Preliminaries

Functions

- o objective
- l_i linear inequality — q_i standard cone
- e_i linear equality — r_i rotated cone

Terminology

- m_c most recently used constraint of type c
- n most recently used variable index
- x vector of original variables
- v vector of all variables

Example

- $l_1(v) \coloneqq 3x_1 - 2,\; l_1(v) \coloneqq l_1(v) + v_3,\;\; l_1(v) \le 0$
  create $3x_1 + v_3 \le 2$

Sum of Norms

Transformation: Utilities

newVar ($\boldsymbol{b}$);

$\boldsymbol{n}$++;
add variable $\boldsymbol{v_n}$;

if ($\boldsymbol{b}$) then add $\boldsymbol{v_n \geq b}$;

newFunc ($\boldsymbol{c}$);

$\boldsymbol{m_c}$++;
add constraint of type $\boldsymbol{c}$;

$\boldsymbol{c_{m_c}(v) \coloneqq 0}$;

Transformation: Sum of Norms

newFunc($\boldsymbol{o}$);
$\boldsymbol{o}_{\boldsymbol{m_o}}(\boldsymbol{v}) \coloneqq \mathbf{0}$;

tranSumNorms(Root, $\boldsymbol{o}_{\boldsymbol{m_o}}(\boldsymbol{v})$, 1);

Transformation: Sum of Norms

```
tranSumNorms ( Node, o(v), c );

case of Node {

 PLUS:   tranSumNorms( Node.left, o(v), c );
         tranSumNorms( Node.right, o(v), c );

 MULT:   tranSumNorms( Node.right, o(v), c·value(Node.left) );

 SQRT:   newVar( 0 );
         o(v) := o(v) + v_n ;

         newFunc( q );
         q_{m_q}(v) := −v_n^2 ;

         tranSumSquares( Node.child, q_{m_q}(v), c );

}
```

Sum of Norms

Transformation: Sum of Squares

```
tranSumSquares ( Node, q(v), c );

case of Node {

 PLUS:   tranSumSquares( Node.left, q(v), c );
         tranSumSquares( Node.right, q(v), c );

 POWER:  newvar(  );
         q(v) := q(v) + v_n^2;

         newfunc( e );
         e_{m_e}(v) := -v_n ;

         tranLinear( Node.left, e_{m_e}(v), c );

 CONST:  q(v) := q(v) + c^2 · value(Node);

}
```

Issues

Which SOCP-solvable forms . . .

- ❖ are of practical use?
- ❖ are worth transforming?
  - ∗ for continuous problems?
  - ∗ for integer problems?