Discovering the Characteristics of Mathematical Programs

John W. Chinneck
Systems and Computer Eng.
Carleton University
Ottawa, Canada

Motivation

- Complex models, large scale:
  - Unexpected results, bad performance, solver failure…
- Limited information returned by (e.g. NLP) solvers:
  - Feasible, KKT conditions satisfied
  - No improvement in many iterations: stopping.
  - Unable to find feasible point.
  - Too many iterations.
  - Various specific failure messages…
- Questions:
  - Why do I have this problem?
  - How do I make the solver run better on this model?
- **Needed:** tools to discover the characteristics of models

Model Characteristics

Some characteristics (e.g. for NLPs):

- Shapes of the constraints and objective (convex, concave, both, almost linear, etc.)
- Shape of the feasible region (convex, non-convex)
- Redundancy of constraints
- Location of feasible region

Insights gained:

- Better understanding of outcomes and behaviour
- Functions that can be approximated (e.g. linear)
- Constraints that can be ignored
- Best type of solution algorithm to apply
- Good starting point

Outline

Theory

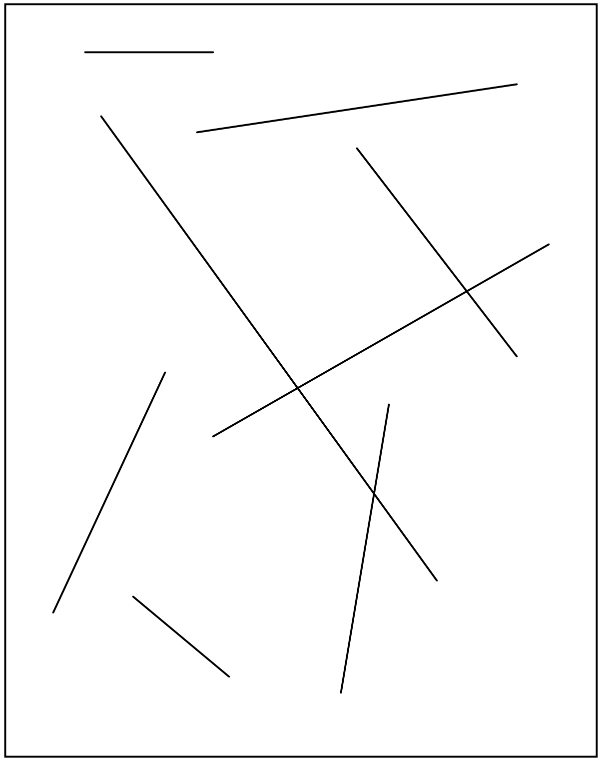
- Sampling for Characteristic Discovery
  - What you can discover
- Tightening Sampling Box for Better Accuracy
- Sampling in Convex Envelopes
  - Hit and run methods
  - Approximating the analytic centre
- Point-Oriented Analysis

Practice

- MProbe software

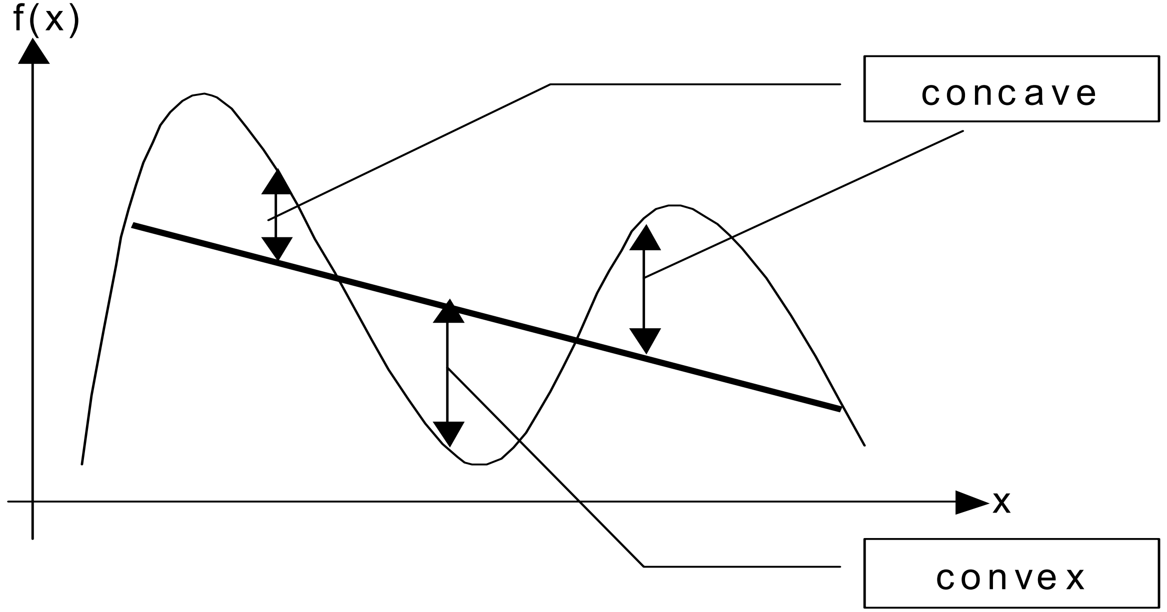
Box Sampling

- Random line segments within variable bounds:
  - Uniform distrn for endpoint 1
  - Uniform distrn for endpoint 2
- Interior points at fixed positions on line segment (e.g. 3 equally spaced)
- Default settings:
  - 500 line segments
  - 3 interior points per line segment
- Info from pts, info from lines



Empirical Function Shape

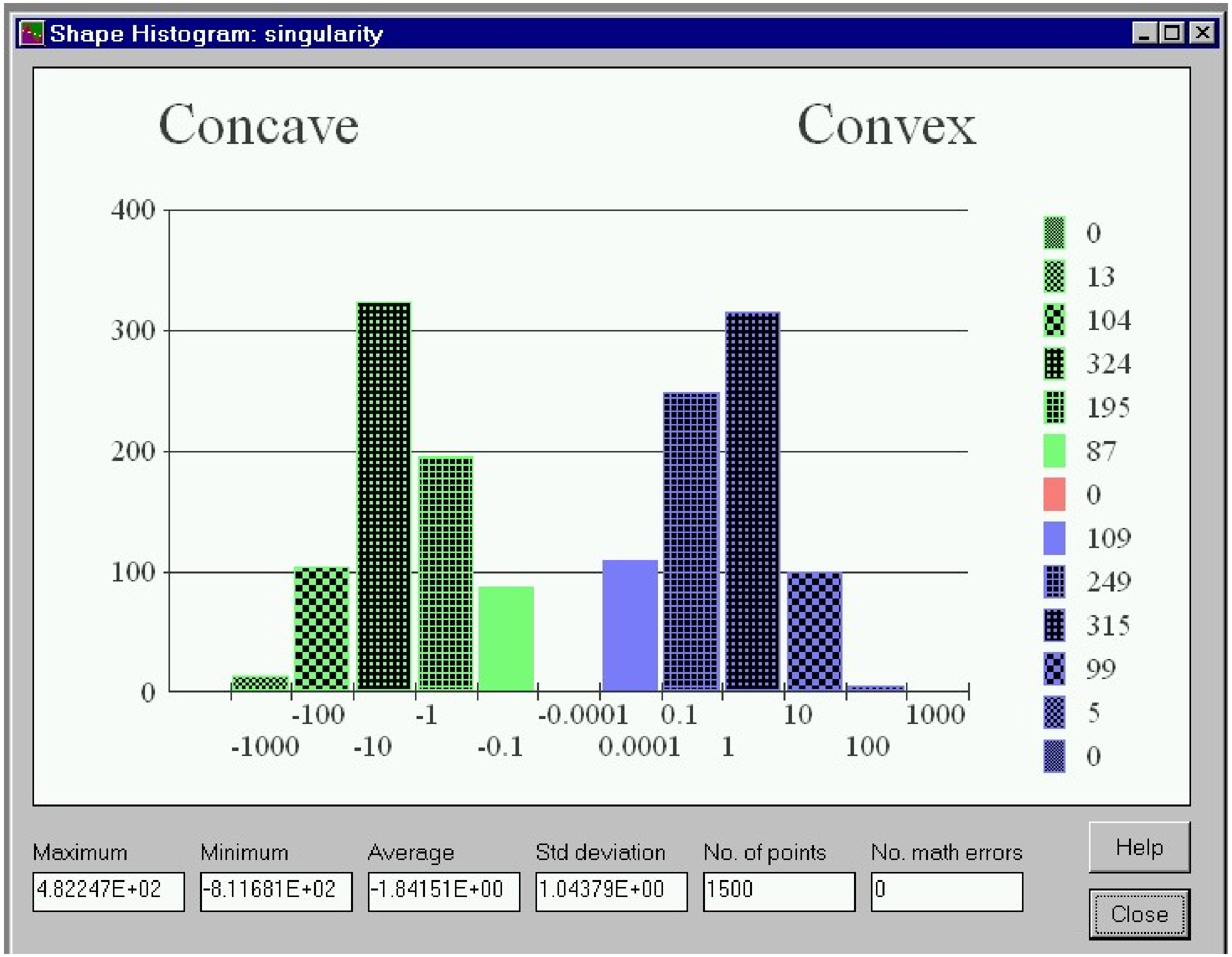
- **Empirical Shape**: based on sampled *differences* between actual and interpolated function values
- Depends on *where* you sample
  - E.g. algebraically nonlinear function may be linear in region of interest



Tolerances

- Difference: interpolated – actual value
- **Equality tolerance**: if difference less than $\pm tol_{equality}$ then interpolated=actual at that point
- **Almost tolerance**: if difference is less than $\pm tol_{almost}$ then interpolated *almost* equals actual at that point
- E.g.: $tol_{equality}$=0.00001, tol_{almost}=0.001
- *Almost* identifies candidates for approximation

Shape Histogram



Function Shape Assessment

linear: all diffs within $\pm tol_{equality}$

convex: all diffs are above $-tol_{equality}$ and at least one above $+tol_{almost}$.

convex, almost linear: all diffs are above $-tol_{equality}$, at least one is between $+tol_{equality}$ and $+tol_{almost}$, and none are above $+tol_{almost}$.

almost convex: at least one diff is between the $-tol_{almost}$ and $-tol_{equality}$, and at least one diff is above $+tol_{almost}$

concave: all diffs are below $+tol_{equality}$, and at least one is below $-tol_{almost}$

concave, almost linear: all diffs are below $+tol_{equality}$, at least one is between $-tol_{equality}$ and $-tol_{almost}$, and none are below $-tol_{almost}$.

almost concave: at least one diff is between $+tol_{almost}$ and $+tol_{equality}$, and at least one diff is below $-tol_{almost}$.

convex and concave: at least one diff is above $+tol_{almost}$, and at least one diff is below $-tol_{almost}$.

convex and concave, almost linear: at least one diff is between $+tol_{equality}$ and the $+tol_{almost}$, and at least one diff is between $-tol_{equality}$ and $-tol_{almost}$.

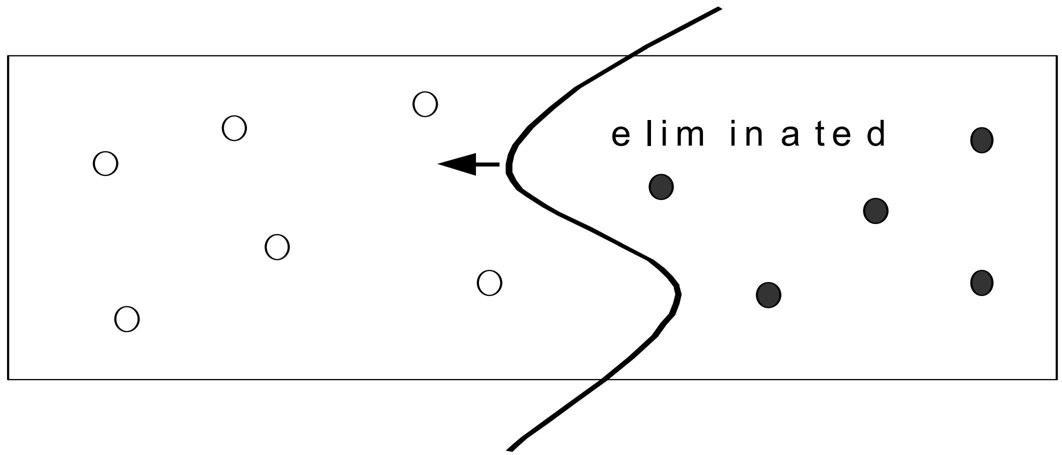
Summary of Constraint Shape Assessments

name	i.d.	bound	alg. shape	emp. shape	reg. effect
myquadratic	0	Less than a constant	Quadratic	Convex	Convex
mynonlinear	1	Greater than a constant	General nonlinear, not quadratic	Convex and concave	Nonconvex
myerrors	2	Less than a constant	General nonlinear, not quadratic	Error	Shape errors
singularity	3	Less than a constant	General nonlinear, not quadratic	Convex and concave	Nonconvex
sinquadratic	4	Equality	General nonlinear, not quadratic	Convex	Nonconvex
impossible	5	Greater than a constant	Quadratic	Convex	Nonconvex
ineffective	6	Greater than a constant	Quadratic	Convex	Nonconvex
ImpossEquality	7	Equality	Quadratic	Convex and concave	Nonconvex
mylinear	8	Less than a constant	Linear	Linear	Convex
multipleChoice	9	Equality	Linear	Linear	Convex
mixedTypes	10	Less than a constant	Linear	Linear	Convex
allInts	11	Interval	Linear	Linear	Convex

(Un)certainty of Empirical Assessments

- *Linear* report: confidence increases with testing
- *Almost linear* reports: confidence increases with testing:
  - Convex, almost linear
  - Concave, almost linear
  - Convex and concave, almost linear
- *Nonlinearity* is certain if report is:
  - Convex
  - Concave
  - Convex and concave
  - xxx, almost linear
- *Convex and concave* is certain if report is:
  - almost convex
  - almost concave
  - convex and concave

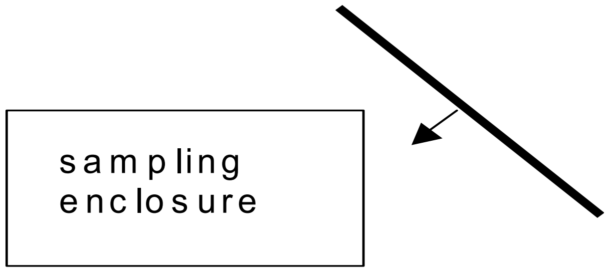
Constraint Effectiveness



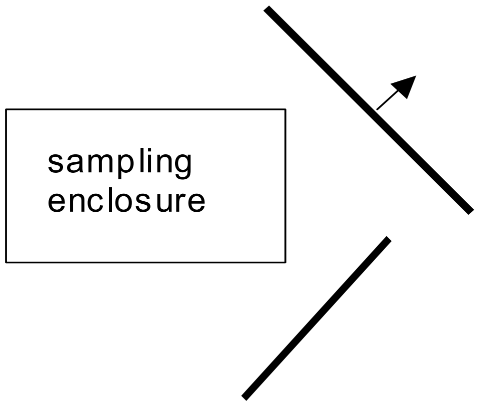
- *Inequalities*: what fraction of the sampling enclosure is eliminated?
- *Equalities*: fractions of sample points above, below, on the function?

Simple Feasibility & Redundancy

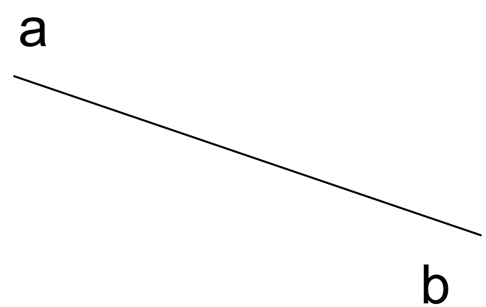
- Simple constraint redundancy (0% effective)



- Simple feasibility test (100% effective)



Function “slope”

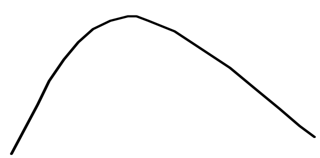


|f(a) – f(b)| / (length a to b)

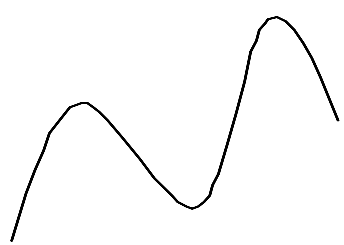
- Multidimensional idea of “steepness”
- Collect in histogram
- Especially useful for objective functions

Objective Optimum Effects

- Objective sense and shape interact:
  - global optimum possible by descent methods
  - Local optimum likely



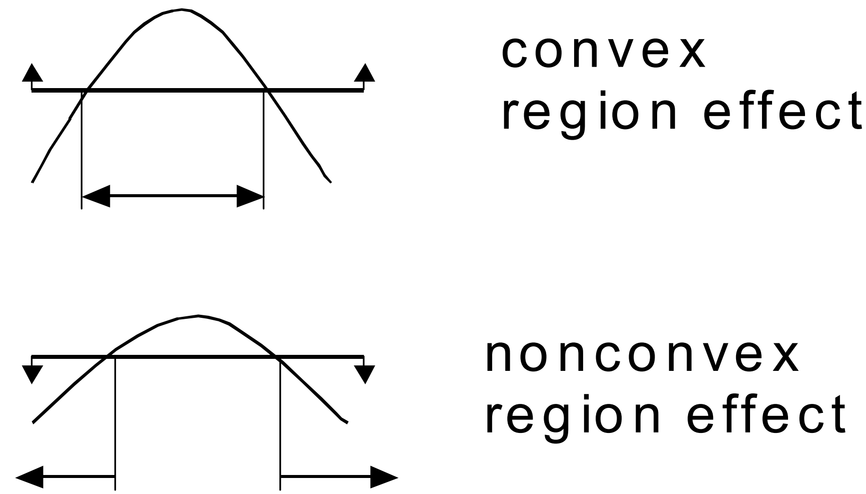
Max: global opt possible
Min: local opt likely



Max or Min: local opt likely

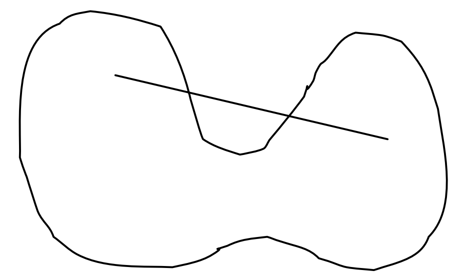
Constraint Region Effect

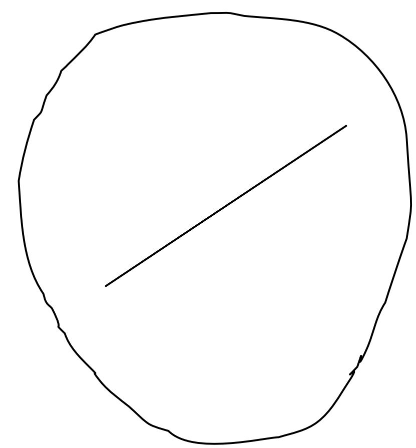
- Effect on shape of constrained region
- *Inequalities*: sense interacts with shape
- *Equalities*:
  – Linear: convex region effect
  – Others: nonconvex region effect



Shape of Constrained Region

- Assessing shape of the constrained region:
  - All constraints have convex region effect: *feasible region is convex* (if it exists)
  - Else *constrained region is nonconvex*
- ***Note***: constraints sampled individually, results compiled

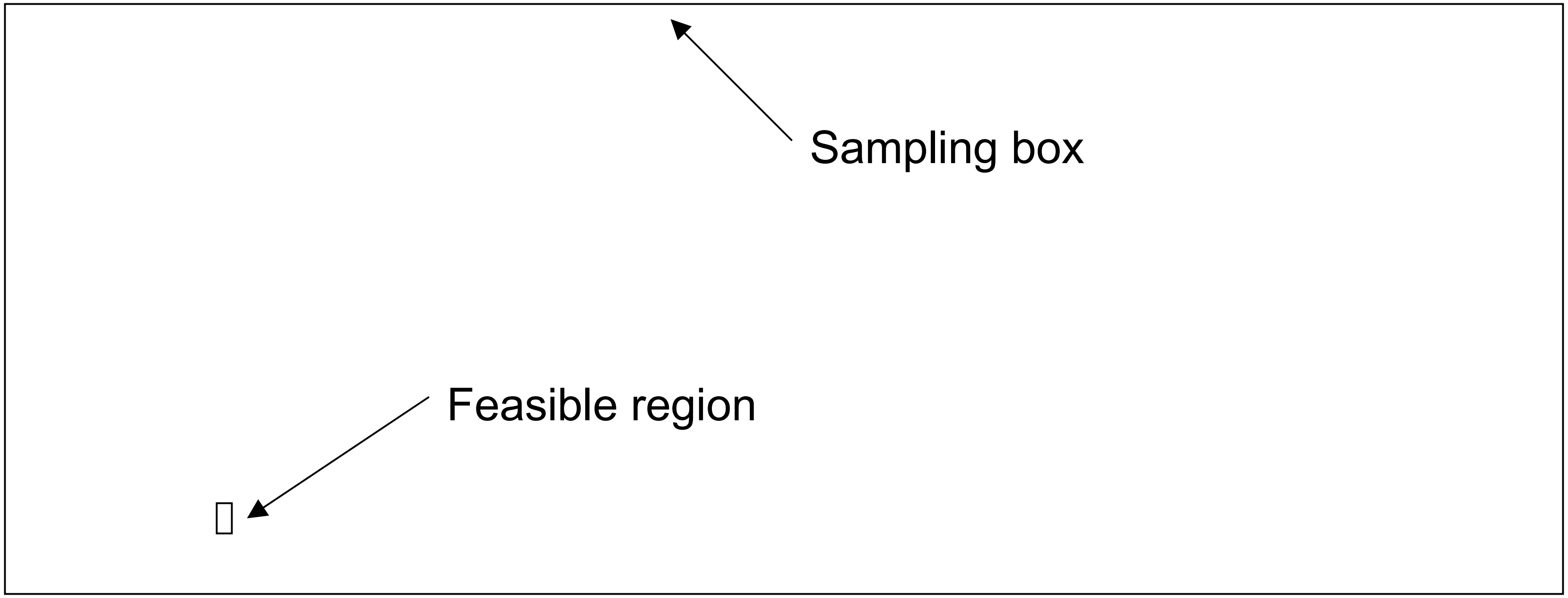




Other Info from Sampling

- Function value statistics
  - histogram, max, min, etc.
- Objective function best sampled value and point (not nec. feasible)
- variables min and max sampled values
- Line segment length
  - effect on conclusions

Tightening Sampling Box for Improved Accuracy



- Sampling box should be a close outer approximation of the feasible region

Tightening Bounds

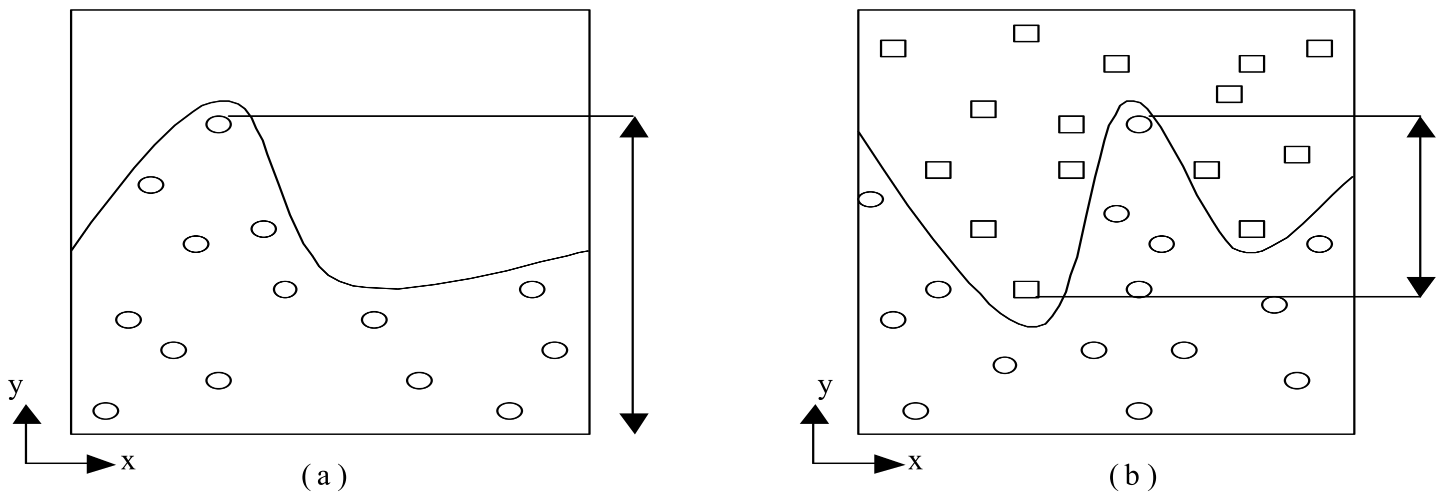
Methods:

- Manual adjustment
- Linear interval analysis
- Nonlinear interval sampling
- Get a nucleus box
- Range cutting
- Constraint Consensus bound tightening
- Max/min sampled values from convex enclosure *(explanation deferred)*

Linear Interval Analysis

- Applies to the subset of linear constraints
- Like standard presolve: bound changes percolate
- E.g.:
  - constraint $2x_1 – 5x_2 \leq 10$ when $-10 \leq x_1, x_2 \leq 10$
  - Tighten x_2 lower bound by applying the constraint when x_1 is at it's lower bound: $2(-10) – 5x_2 \leq 10 \Rightarrow x_2 \geq -6$.
  - Conclusion: true bounds are $-6 \leq x_2 \leq 10$.

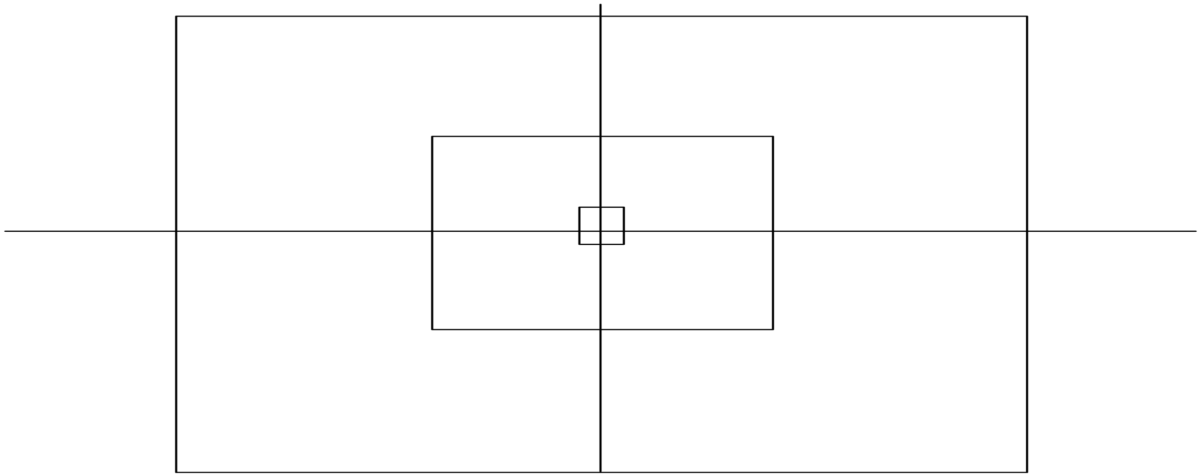
Nonlinear Interval Sampling



a) inequality b) equality

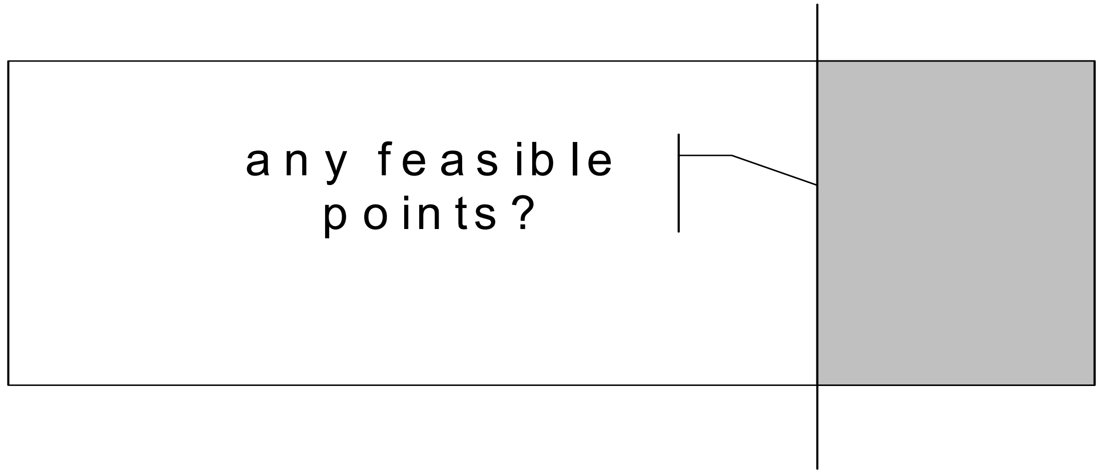
- Apply to each constraint in turn
- overtightens
- non-overlap? return the gap itself

Get a Nucleus



- For unbounded variables
- Look at constraints involving the variable that were never satisfied during interval sampling
- try gradually larger boxes centred at origin. Stop when next box shows no feasible points

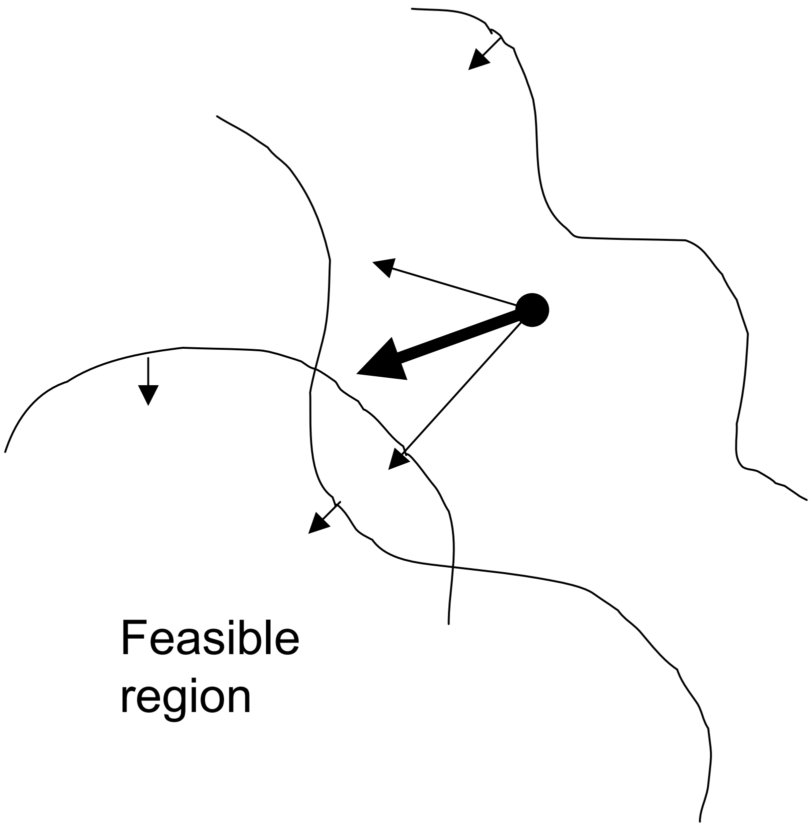
Nonlinear Range Cutting



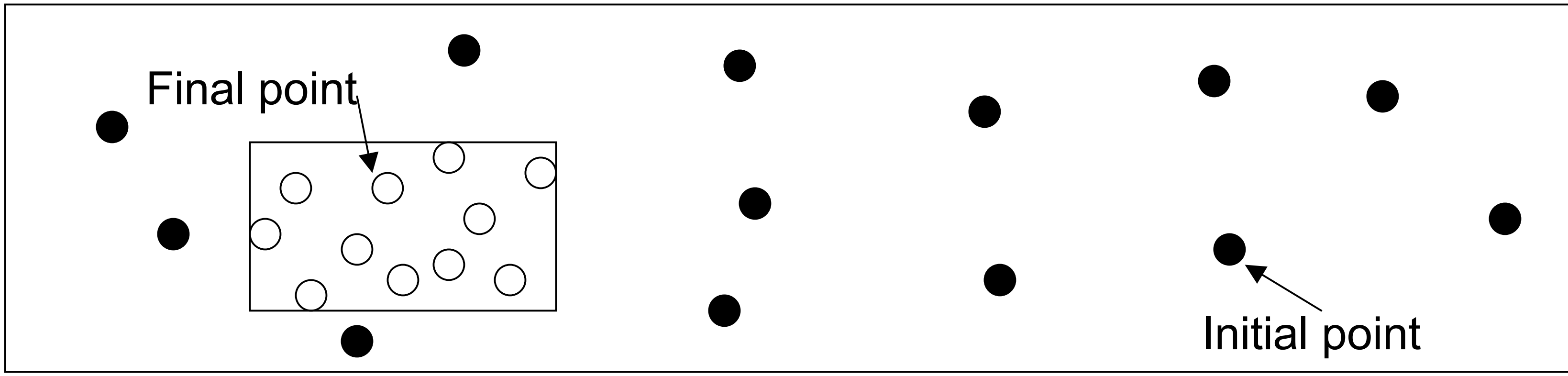
- Accept cut if at least one constraint never satisfied when sampling in the zone
- *equalities*: "satisfied" if find one pt ≤ rhs and one pt ≥ rhs

Constraint Consensus Heuristic

- Quickly achieves approximate feasibility
- For each violated constraint: estimate vector to achieve feasibility
- Consensus vector: Component-wise average of feasibility vectors
- Update point using consensus vector
- Repeat until close to feasible



Constraint Consensus bound tightening



- Apply CC method from numerous random initial points in current sampling box
- Shrink bounds to encompass cloud of final points

Sampling in Convex Enclosures

How do you find a convex region that encloses the feasible region?

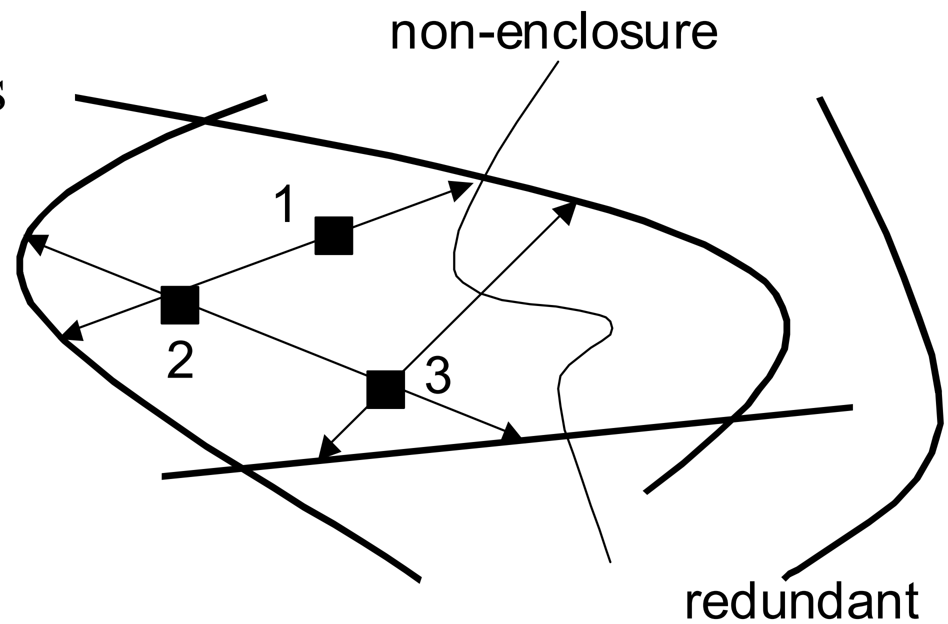
Procedure:

1. analyze constraint region effects by box sampling
2. select inequalities that have convex region effects and all variable bounds

Sample via hit-and-run methods

Hit-and-Run Sampling

- hit constraints are *necessary;* unhit constraints are *redundant* (relative to enclosure)
- estimate fraction of enclosure surface area
- non-enclosure constraints sampled as usual (shape, effectiveness, etc.)



Finding an Initial Feasible Point

Need an initial feasible point to start hit-and-run

Method 1:

- Sample randomly until at least one constraint satisfied
- Thereafter use hit-and-run to <u>keep</u> constraints satisfied
- Stop when all constraints satisfied.
- ***Note:*** bias hit-and-run sampling rays according to variable bounds (long thin boxes are a problem)

Method 2:

- Apply constraint consensus method

The Analytic Centre

Analytic centre: ***P*** is point that maximizes
W=∑[*ln(distance to constraint)* over all constraints.

- Best place to launch rays for estimating surface frac.

Heuristic for finding analytic centre:

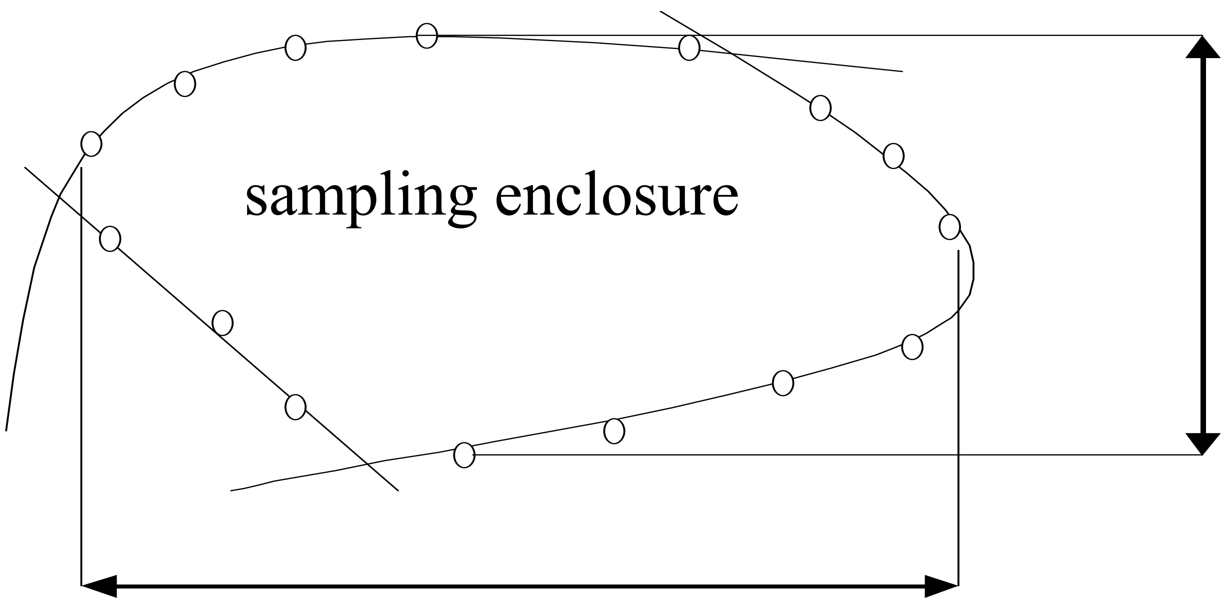
- Initial feasible pt is first estimate of ***P***. *W* always calculated over necessary constraints found so far.
- Launch hitting rays from ***P***.
- Get new hit-and-run launch point ***x***.
- If *W*(***x***)>*W(**P**)* then ***P***=***x***.

Advantages:

- Pushes ***P*** away from discovered necessary constraints towards undiscovered necessary constraints
- Quick convergence to analytic centre.

“replace current bounds with max/min sampled values”

- After convex enclosure sampling: tighten bounds
- Use hit points to (over)tighten the variable bounds



Point-Oriented Analysis

- Finding (near) feasible points
  - *What is a good starting point?*
- Finding (near) optimal points
- Analyzing features of points
  - *Why did my solver stop here?*

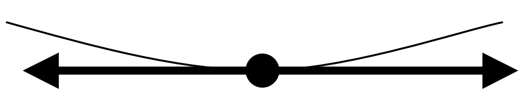
Finding (Near) Feasible and Optimum Points

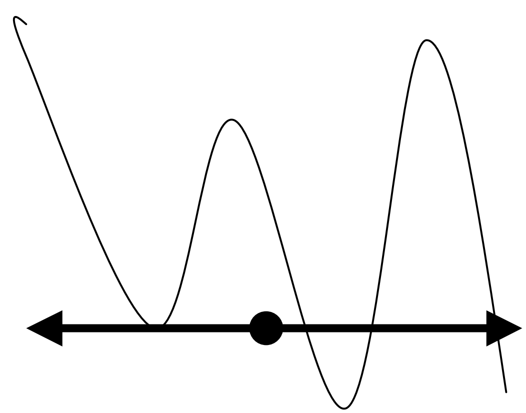
- ***Finding a near-feasible point***:
  - constraint consensus method
- ***Finding a near-optimal point***:
  - keep track of feasibility status and objective function value of all sampled points.
  - Note *best found point* and *best found feasible point*

Analyzing Point Features

- Create a small box around a point:
  - E.g. shrink bounds to 1% of each current edge dimension, centred around point
- Look at objective “flatness” in the box
  - Histogram of objective “slope” in the box.

Flat (1-D)





MProbe

- Software tool embodying these and other analytic methods
- Reads AMPL, GAMS, MPS files
- Essential part of an *integrated development environment* for math programming

MProbe Workshops

Variables Workshop
- Shift, tighten variable bounds

Constraints and Objectives Workshops
- Analyze shape, effectiveness, redundancy, set up convex enclosures for tighter sampling

Constrained Region Workshop
- Analyze shape of feasible region

Points Workshop
- Exchange points with solvers, look for near-feasible points, etc.

Additional Features

- Model statistics
- “Snapping” of integer/binary variables
- Spreadsheet-like displays of constraints, objectives, variables
  - View subsets by category (e.g. only nonlinear constraints, only binary variables)
  - Sort on any column
  - Navigate (e.g. see all constraints that contain variable *x*)
- User adjustment of tolerances, histogram cells
- Text file trace of session
- Help system

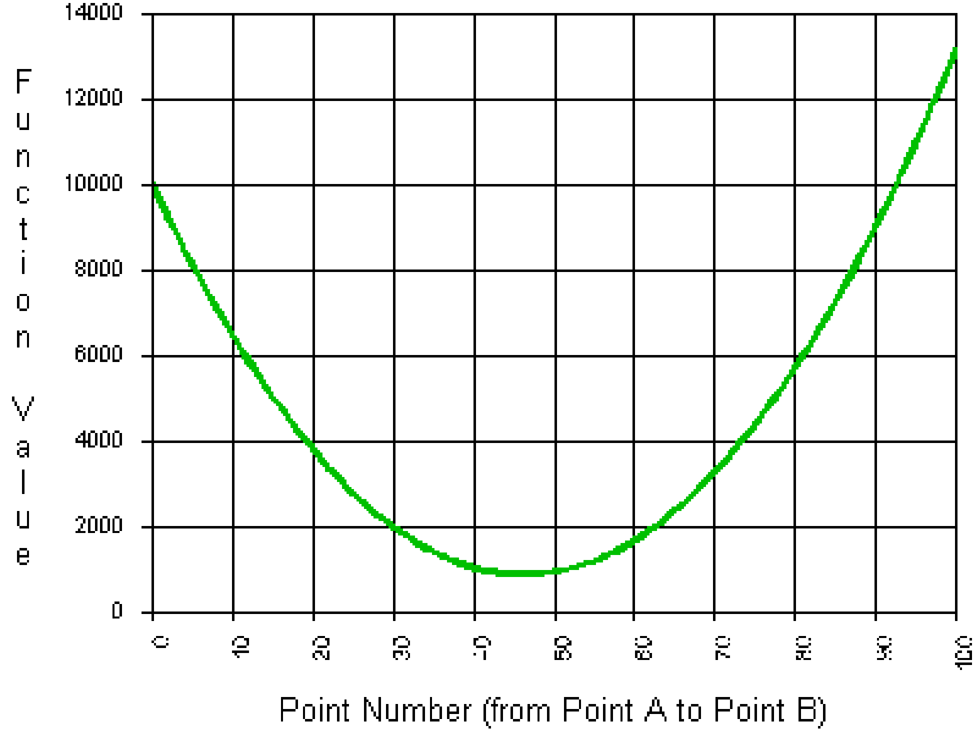
Constraints Workshop

Show only > | all | ☐ Reverse selection | 12 of 12 constraints visible

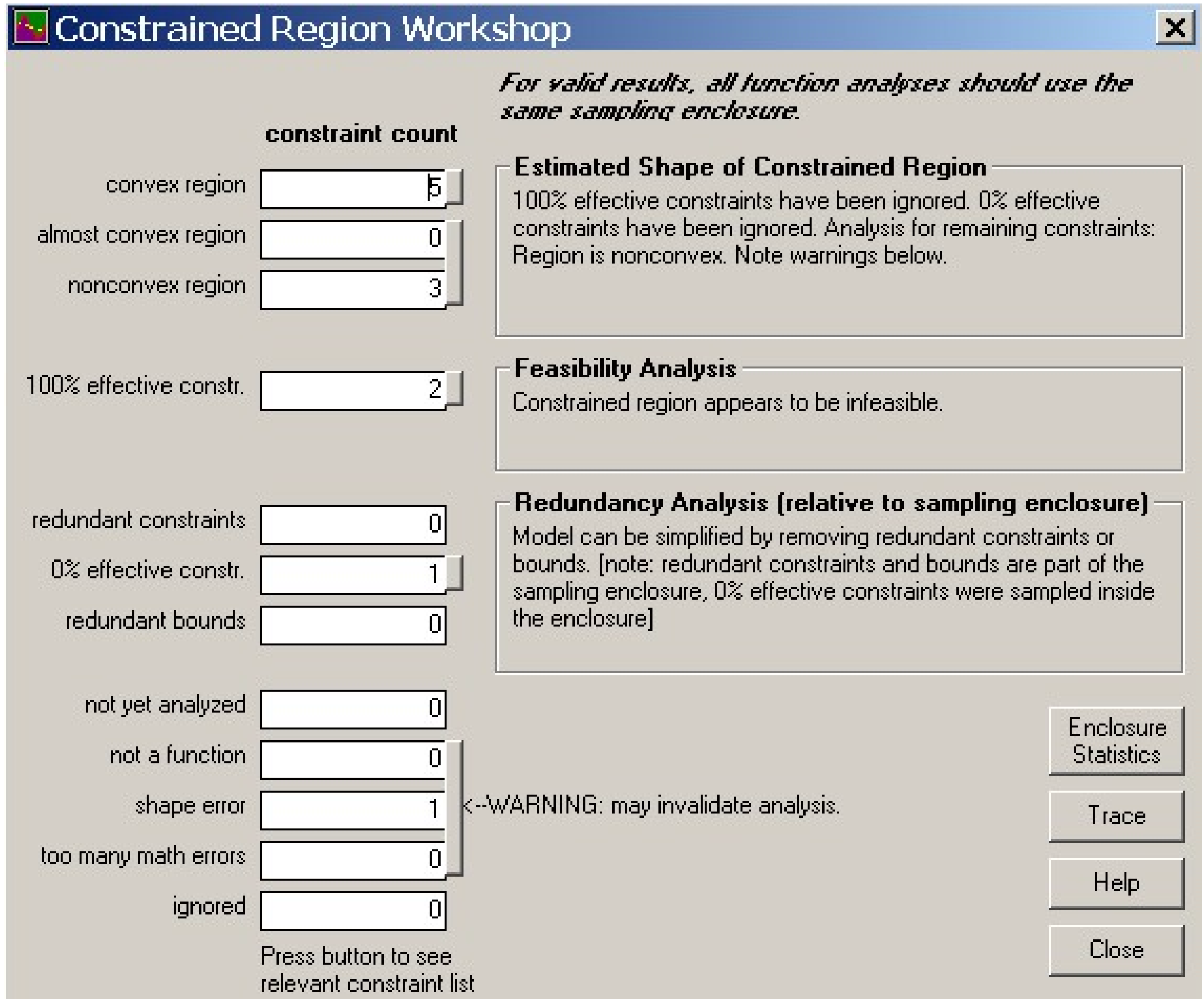
name	i.d.	bound	alg. shape	emp. shape	reg. effect	effectiveness	tot. var	real va	int+bin	int. var.	bin. var	sampling	redundancy	surface
myquadratic	0	Less than a	Quadratic	Convex	Convex	n.a.	2	2	0	0	0	Enclosure	Necessary	0.165
mynonlinear	1	Greater tha	General nonlin	Convex and concave	Nonconvex	0.5120	2	2	0	0	0	Analyze	n.a.	0
myerrors	2	Less than a	General nonlin	Convex and concave	Nonconvex	0.1974	2	2	0	0	0	Analyze	n.a.	0
singularity	3	Less than a	General nonlin	Almost convex	Almost convex	0.0000	2	2	0	0	0	Analyze	n.a.	0
sinquadratic	4	Equality	General nonlin	Convex	Nonconvex	Possible. LT:0.	2	2	0	0	0	Analyze	n.a.	0
impossible	5	Greater tha	Quadratic	Convex	Nonconvex	1.0000	1	1	0	0	0	Analyze	n.a.	0
ineffective	6	Greater tha	Quadratic	Convex	Nonconvex	0.0000	2	2	0	0	0	Analyze	n.a.	0
ImpossEquality	7	Equality	Quadratic	Convex and concave	Nonconvex	1.0000	4	2	2	0	2	Analyze	n.a.	0
mylinear	8	Less than a	Linear	Linear	Convex	n.a.	2	2	0	0	0	Enclosure	Necessary	0.043
multipleChoice	9	Equality	Linear	Linear	Convex	Possible. LT:0.	3	0	3	0	3	Analyze	n.a.	0
mixedTypes	10	Less than a	Linear	Linear	Convex	n.a.	3	1	2	1	1	Enclosure	Necessary	0.099
allInts	11	Interval	Linear	Linear	Convex	n.a.	3	0	3	3	0	Enclosure	Necessary	0.183

Visualizing Shape: Function Profile

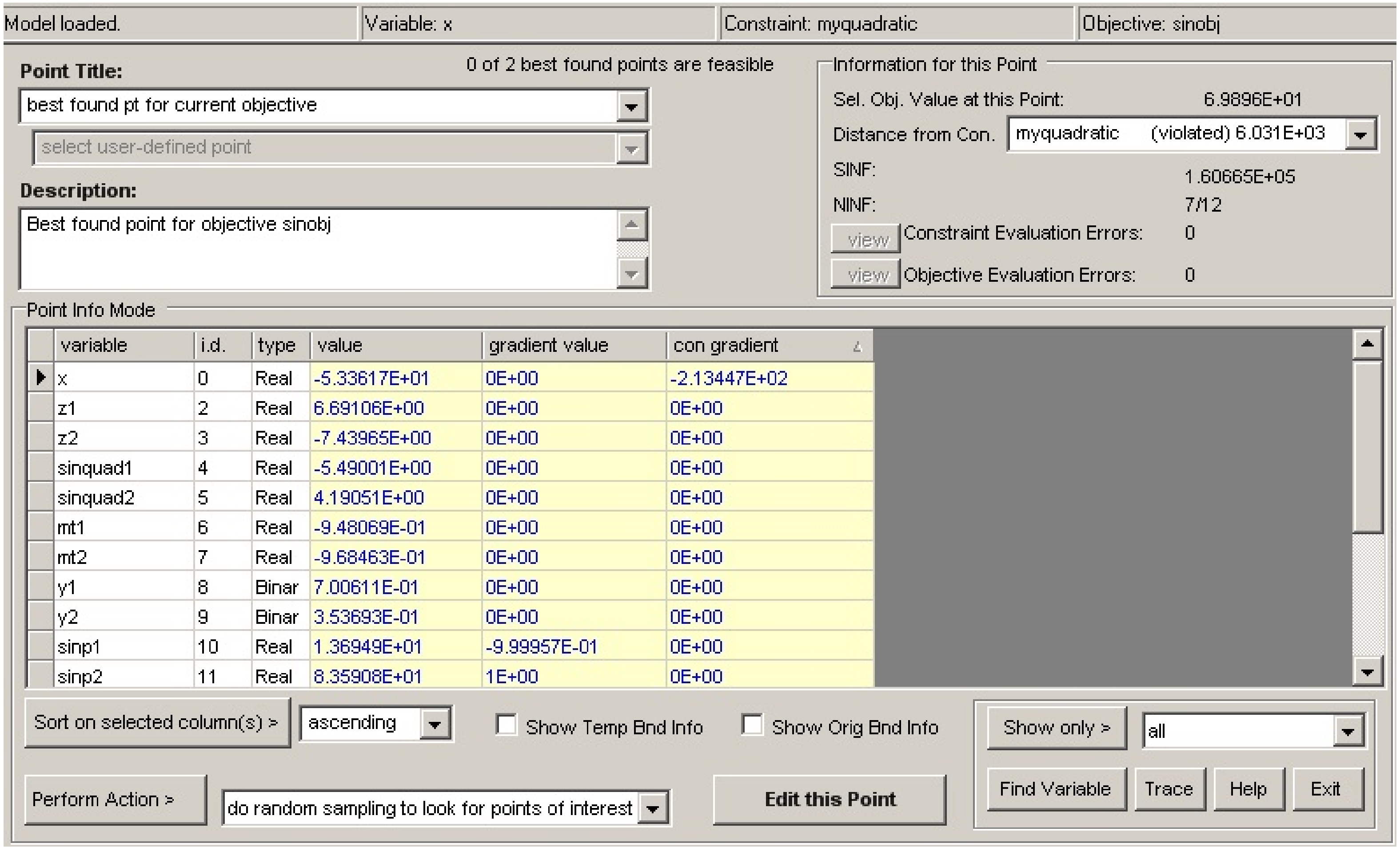
- 2 dimensional plot between 2 endpoints in n-space
- End-points selectable and configurable in multiple ways



Constrained Region Workshop



Points Workshop



Conclusions

- Model probing and analysis tools a vital part of IDE for math programming
- Shape analysis tools are heuristic and based on random sampling
  - Tools don't *always* work, but *often* do.
  - Can be slow for very large or very complex models
  - Performance depends on characteristics of the model
- Download: www.sce.carleton.ca/faculty/chinneck/mprobe.html

Research Directions

- Better Sampling
  - Interior pts on line segments means more samples towards centre of sampling box
- Better bound tightening
  - Better interaction among the component methods
- Determining best approximations of functions (e.g. best piecewise linear approximation)
- Connection to computer algebra system (Matlab?)
- Tool for identifying implied equalities
- Etc……….

References

- Chinneck, J.W. (2003), “The Constraint Consensus Method for Finding Approximately Feasible Points in Nonlinear Programs”, *INFORMS Journal on Computing*, to appear.
- Chinneck, J.W. (2002), “Discovering the Characteristics of Mathematical Programs via Sampling”, *Optimization Methods and Software,* vol. 17, no. 2, pp. 319-352.
- Chinneck, J.W. (2001), “Analyzing Mathematical Programs using MProbe”, *Annals of Operations Research*, vol. 104, pp. 33-48.