Introduction to Nonlinear Optimization

GIAN Short Course on Optimization: Applications, Algorithms, and Computation

Sven Leyffer

Argonne National Laboratory

September 12-24, 2016

Outline

Objective Function and Constraints

Optimization

Art of finding a best solution from collection alternatives.

Everyone optimizes: application in ...

- Science: design of experiments
- Engineering: power-grid control and design
- Finance: pricing of options, optimal portfolio selection.
- Medicine: optimal radiation dose design
- Economics: optimal transition to clean energy
- Big data: machine learning ... training of neural nets

... more details tomorrow.

The Most Important Slide of the Course

The Most Important Slide of the Course

Please Ask Questions!!!

- There are no stupid question …
  … there are only stupid teachers!
- If **YOU** have a question, then **YOUR neighbor** has the same!
- If **you all ask** question, then **I know** you are interested!

Objective Function and Constraints

Ingredients of Optimization

- **Decision Variables,** x**,** model decisions.
- **Constraints** model acceptable values of x.
- **Objective(s)** model our goals / performance measure.

$$
\begin{array}{llll}
\underset{x}{\text{minimize}} & f(x) & & \text{objectve function} \\
\text{subject to} & l_c \leq c(x) & \leq u_c & \text{nonlinear constraints} \\
& l_A \leq A^T x & \leq u_A & \text{linear constraints} \\
& l_x \leq \quad x & \leq u_x & \text{simple bounds} \\
& & x \in \mathcal{X} & \text{structural constraints}
\end{array}
$$

Objective Function and Constraints

$$\begin{array}{llll} \underset{x}{\text{minimize}} & f(x) & & \text{objective function} \\ \text{subject to} & l_c \leq c(x) & \leq u_c & \text{nonlinear constraints} \\ & l_A \leq A^T x & \leq u_A & \text{linear constraints} \\ & l_x \leq \quad x & \leq u_x & \text{simple bounds} \\ & & x \in \mathcal{X} & \text{structural constraints} \end{array}$$

Basic Blanket Assumptions

We make the following blanket assumptions:

1. $x \in \mathbb{R}^n$ finite dimensional.
2. Functions, $c : \mathbb{R}^n \to \mathbb{R}^m$ and $f : \mathbb{R}^n \to \mathbb{R}$ are smooth.
3. Bounds, $l_c, u_c, l_A, u_A, l_x, u_x$ can be infinite.
4. Set $\mathcal{X} \subset \mathbb{R}^n$ imposes structural restrictions x (later).

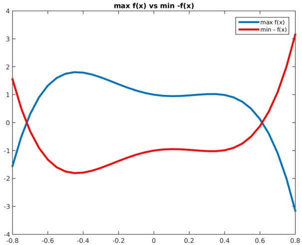
Objective Function and Constraints

$$\begin{array}{llll} \underset{x}{\text{minimize}} & f(x) & & \text{objective function} \\ \text{subject to} & l_c \le c(x) & \le u_c & \text{nonlinear constraints} \\ & l_A \le A^T x & \le u_A & \text{linear constraints} \\ & l_x \le \quad x & \le u_x & \text{simple bounds} \\ & & x \in \mathcal{X} & \text{structural constraints} \end{array}$$

To Minimize or To Maximize?

$\max f(x)$ equivalent to $-\min(-f(x))$

... wlog only consider minimization



Notation

- Subscripts denote components of (column) vectors:

$$a \in \mathbb{R}^n \quad \text{has components} \quad a = (a_1, \ldots, a_n)^{\mathsf{T}}.$$

- For $a, b \in \mathbb{R}^n$ vectors: $a \leq b$ means that $a_i \leq b_i \ \forall i$.
- Use upper case letters for matrices:

$$A \in \mathbb{R}^{n \times m},\ x \in \mathbb{R}^n \quad \text{then} \quad \left[A^T x\right]_i = \sum_{j=1}^{n} \left[A^T\right]_{ij} x_j = \sum_{j=1}^{n} A_{ji} x_j$$

- Calligraphic type indicates finite or infinite sets, e.g.

$$\mathcal{X} \subset \mathbb{R}^n, \quad \text{or} \quad \mathcal{A} \subset \{1, \ldots, n\}.$$

Programming vs. Optimization

Optimization Problems also called "Program" ... WWII.

Example: Design of Reinforced Concrete Beam

- **Variables:**
  - x_1 = area of re-inforcement,
  - x_2 = width of beam,
  - x_3 = depth of beam.
- **Objective:** minimizing cost of reinforced beam
- **Constraints:**
  - Support minimum amount of load.
  - Bounds on width/depth ratio and variables (positivity).

Example: Design of Reinforced Concrete Beam

- **Variables:**
  - $x_1 =$ area of re-inforcement, e.g. $x_1 \in \{40, 45, \dots, 75\}$
  - $x_2 =$ width of beam,
  - $x_3 =$ depth of beam.
- **Objective:** minimizing cost of reinforced beam
- **Constraints:**
  - Support minimum amount of load.
  - Bounds on width/depth ratio and variables (positivity).

$$\begin{array}{lll}
\underset{x}{\text{minimize}} & f(x) = 29.4x_1 + 0.6x_2x_3 & \text{cost of beam} \\
\text{subject to} & c(x) = x_1x_2 - 7.735\frac{x_1^2}{x_2} \geq 180 & \text{load constraint} \\
& x_3 - 4x_2 \geq 0 & \text{width/depth ratio} \\
& 40 \leq x_1 \leq 77,\ x_2 \geq 0,\ x_3 \geq 0 & \text{simple bounds,}
\end{array}$$

In practice, area of reinforcement, x_1, is discrete ... include in $\mathcal{X}$.

Outline

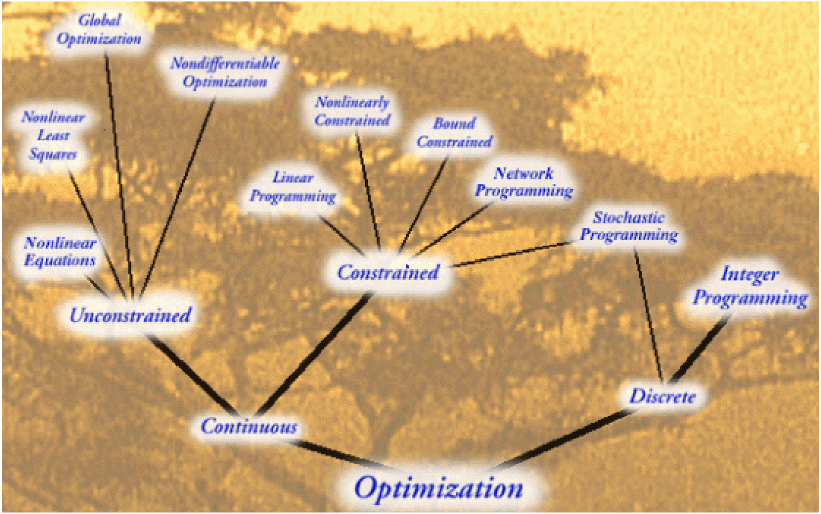
Classification of Optimization Problems

$$\begin{array}{llll} \underset{x}{\text{minimize}} & f(x) & & \text{objective function} \\ \text{subject to} & l_c \leq c(x) & \leq u_c & \text{nonlinear constraints} \\ & l_A \leq A^T x & \leq u_A & \text{linear constraints} \\ & l_x \leq \;\; x & \leq u_x & \text{simple bounds} \\ & & x \in \mathcal{X} & \text{structural constraints} \end{array}$$

Classify optimization problems by

- Type/class of objective function(s).
- Type/class of constraint functions.
- Structure of constraints like $A^T x$.
- Type of variables.

NEOS Optimization Tree



Classification by Type of Constraint

Assume $\mathcal{X} = \mathbb{R}^n$, and $f(x)$, $c(x)$ twice continuously differentiable

- **Unconstrained Optimization** all $x \in \mathbb{R}^n$ feasible:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \;\; f(x).$$

Special Case: Least-Squares Problem

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \;\; f(x) = \sum_{j=1}^{m} \left(r_j(x)\right)^2 ,$$

- **Bound Constrained Optimization** only bounds constraints:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \;\; f(x) \quad \text{subject to } l \le x \le u,$$

where $l, u \in \mathbb{R}^n$ can be infinite.
Special case: $f(x) = c^T x$ solved trivially.

... studied in Part II of this course.

Classification by Constraint Type

Assume $\mathcal{X} = \mathbb{R}^n$, and $f(x)$, $c(x)$ twice continuously differentiable

- **Linearly Constrained Optimization** nonlinear objective and linear constraints

$$\begin{array}{lll} \underset{x}{\text{minimize}} & f(x) & \text{objective function} \\ \text{subject to} & l_A \leq A^T x \leq u_A & \text{linear constraints} \\ & l_x \leq \quad x \quad \leq u_x & \text{simple bounds} \end{array}$$

Important Special Cases:

- **Linear Programming** Objective function is linear:

$$f(x) = c^T x$$

- **Quadratic Programming** Objective function is quadratic:

$$f(x) = x^T G x / 2 + g^T x + a$$

wlog assume $a = 0$ Why???

... studied in in Part III of this course.

Classification by Constraint Type

Assume $\mathcal{X} = \mathbb{R}^n$, and $f(x)$, $c(x)$ twice continuously differentiable

- **Equality Constrained Optimization** all constraints are equations:

$$\begin{array}{l} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x) \\ \text{subject to } c(x) = 0. \end{array}$$

Special Case: Only linear equality constraints: $A^T x = b$.

... studied in Part III of this course.

Classification by Constraint Type

Assume $\mathcal{X} = \mathbb{R}^n$, and $f(x)$, $c(x)$ twice continuously differentiable

- **Nonlinearly Constrained Optimization**

$$
\begin{array}{llll}
\underset{x}{\text{minimize}} & f(x) & & \text{objective function} \\
\text{subject to} & l_c \leq c(x) & \leq u_c & \text{nonlinear constraints} \\
& l_A \leq A^T x & \leq u_A & \text{linear constraints} \\
& l_x \leq \quad x & \leq u_x & \text{simple bounds} \\
& & x \in \mathcal{X} & \text{structural constraints}
\end{array}
$$

Programming vs. Optimization

This problem is also called a Nonlinear Programming Problem.

Classification by Type of Variables

Variable type is encoded in $x \in \mathcal{X}$:

- **Continuous Variables** are variables with $x \in \mathbb{R}^n$
  ... leverage classical calculus.
- **Discrete Variables** $\mathcal{X}$ is discrete subset:
  - *Binary Variables* $\mathcal{X} = \{0, 1\}^n$ model logic.
  - *Integer Variables* $\mathcal{X} = \mathbb{Z}^n$ model numbers of equipment.
  - *Discrete Variables* from discrete set, e.g.
    $\mathcal{X} = \{1/4, 1/2, 1, 2, 4, ...\}$
    ... can be modeled with binary variables.

$\Rightarrow$ Integer or discrete programming problems

Often have mixture of continuous and discrete variables, called **mixed-integer programs** (MIPs).

... study MIPs in Part IV of this course.

Classification by Type of Variables

Additional classes of variables:

- **State and Control Variables** arise in control problems:
    - Infinite-dimensional variables.
    - $x(t)$ control, or $u(t, x, y, z)$ PDE-constrained optimization.
- **Random Variables** arise in robust or stochastic optimization: also called second-stage variables ... optimize expect cation

Infinite-dimensional optimization problems ... over function spaces.

Must be discretized on mesh, or by drawing random samples
$\Rightarrow$ discretized problem is standard NLP.

Classification by Type of Variables

New classes of constraints have emerged in practical applications:

- **Semi-Definite Optimization** involve matrix variables: $X \in \mathbb{R}^{n \times n}$, such that X positive semi-definite (psd): $X \succeq 0$.

Recall Positive Definiteness

A symmetric matrix $X \in \mathbb{R}^{n \times n}$ is psd, iff all eigenvalues are nonnegative.

- **Second-Order Cone Constraints** special class of quadratic constraint:

$$\left\{ (x_0, x) \in \mathbb{R} \times \mathbb{R}^n \mid x_0 \geq ||x||_2 \right\},$$

also known as the ice-cream cone.

... constraints generalize nonnegativity, and form a cone.

Classification by Functional Forms

Distinguish problems by their functional forms:

- **Smooth** $f(x)$, $c(x)$ twice continuously differentiable
- **Nonsmooth** if $f(x)$ or $c(x)$ Lipschitz continuous:
  e.g. $f(x) = \|A^T x - b\|_2^2 + \|x\|_1$ in compressed sensing
- **Multi-Objective Optimization** more than one goal:

$$f(x) = (f_1(x), \ldots, f_q(x))$$

  models trade-offs (fuel consumption vs. take-off weight).
  Can be transformed into single-objective using:

$$\underset{x}{\text{minimize}} \sum_{i=1}^{q} w_i f_i(x) \quad \text{for some weights} \quad w_i \geq 0$$

... here, concentrate on smooth, single-objective problems.

Outline

Optimization Software Eco-System

Optimization Solvers

- Commercial Packages: Cplex, GuRoBi, XPRESS, Knitro, ...
- Open-Source Solvers: Cbc, Scip, Ipopt, Minotaur, ...

... written in C/C++, Fortran, ... interface to Python, Matlab.

Optimization Modeling Languages

- Express optimization problems in high-level language.
- Interfaces to commercial & open-source solvers
- Commercial Languages: AMPL, GAMS,
- Open-Source Languages: Zimpl, JuMP (Julia for MP), ...

... easy & efficient modeling of various optimization problems.

Introduction to AMPL Modeling Language

We have a full (temporary) license of AMPL for the course

Optimization Problem

$$\min_{x} \exp(-x_1) + \sum_{i=2}^{3} x_i^2$$

$$\text{s.t. } \; x_1 \log(x_2) + x_2^3 \geq 1$$

$$5 \geq x_1, x_2, x_3 \geq 0$$

AMPL Formulation

```
var x{1..3} >=0, <=5; # ... variables

minimize              # ... objective functn
 f: exp(-x[1]) + sum{i in 2..3} x[i]^2;

subject to            # ... constraints
 con: x[1]*log(x[2]) + x[2]^3 >= 1;
```

Beware: $x_2 > 0$ ... $\log(x_2)$ undefined for $x_2 \leq 0$!

Running & Trouble Shooting an AMPL Model

1. Create a `*.mod` model file (see file)
2. Start `ampl`; load model (e.g. `Model1.mod`); select solver:

```
ampl: reset; model Model1.mod;
ampl: option solver ipopt;
ampl: solve;
```

Running & Trouble Shooting an AMPL Model

1. Create a `*.mod` model file (see file)
2. Start `ampl`; load model (e.g. `Model1.mod`); select solver:

```
ampl: reset; model Model1.mod;
ampl: option solver ipopt;
ampl: solve;
```

3. Display the answer or trouble shoot

```
ampl: display _varname, _var.lb, _var, _var.ub;
ampl: display _conname, _con.lb, _con.body, _con.ub;
ampl: expand;
```

... list variable/constraint name, lower bnd, body, upper bnd

... shows all constraints and objective functions

Running & Trouble Shooting an AMPL Model

1. Create a `*.mod` model file (see file)
2. Start `ampl`; load model (e.g. `Model1.mod`); select solver:

```
ampl: reset; model Model1.mod;
ampl: option solver ipopt;
ampl: solve;
```

3. Display the answer or trouble shoot

```
ampl: display _varname, _var.lb, _var, _var.ub;
ampl: display _conname, _con.lb, _con.body, _con.ub;
ampl: expand;
```

   ... list variable/constraint name, lower bnd, body, upper bnd
   ... shows all constraints and objective functions

4. We forgot to ensure $x_2 > 0$ so that $\log(x_2)$ defined:

```
ampl: let x[2] := 1;
ampl: solve;
```

   ... assigns an initial value to x_2 (different from default, 0).

Other Components of an AMPL Model

- We can define sets in AMPL & have set operations

```
set Orig  := { 'ORD','HAM','TXL','BOM', 'JLR' };
set Dest  := { 'ORD','HAM','TXL','BOM', 'JLR' };
set Trips within Orig cross Dest;
```

  - Defines sets of origins, $\mathcal{O}$, destinations, $\mathcal{D}$
  - Defines subset and Trips, $\mathcal{T} \subset \mathcal{O} \times \mathcal{D}$
  - Now we can fill Sven's travel itinerary

```
let Trips := { ('ORD','BOM'), ('BOM','JLR'),
               ('JLR','BOM'), ('BOM','ORD'),
               ('ORD','TXL'), ('HAM','ORD') };
```

  ... clearly Sven is maximizing discomfort!

- We can define parameters (constants) with attributes:

```
param N integer, >0, default 32;
param h := 1/N;
```

... means that `N` is a positive integer with default value 32.

Model and Data Files

Often run same model with different data

Model & Data Files

- Model files define the structure of the problem.
- Data files define the data/instance of the problem.

- Example, the famous diet problem

```
reset; model diet.mod; data diet.dat; solve;
```

  ... loads a problem file, it's data, and solves it.
- Run files can be useful, e.g. `diet.ampl`:

```
reset; model diet.mod; data diet.dat; solve;
display Buy, Diet;
```

  ... can add `for` loops, etc.

Model and Data Files

Often run same model with different data

Model & Data Files

- Model files define the structure of the problem.
- Data files define the data/instance of the problem.

- Example, the famous diet problem

```
reset; model diet.mod; data diet.dat; solve;
```

  ... loads a problem file, it's data, and solves it.

- Run files can be useful, e.g. `diet.ampl`:

```
reset; model diet.mod; data diet.dat; solve;
display Buy, Diet;
```

  ... can add `for` loops, etc.

Best way to learn computing is to code!

Good Coding Practices (Language Independent)

Good Coding Practices

Good coding practices make for better software.

- Use consistent naming scheme
  (e.g. Upper case for sets, lower case for variables, ...)
- Use consistent indentation
- Add lot's of comments to the program!
- Consider using `CamelCode` to name variables etc.
- Limit line length.
- Organize your files and folders consistently.

... it all helps YOU understand YOUR code in 2 months!

Good Coding Practices

Bill Gropp on Writing Code

Take pride in your software!

- University of Illinois Urbana-Champaign, Thomas M. Siebel Chair in CS
- Awarded Blue Waters Professorship
- Acting Director of National Center for Supercomputing Applications (NCSA)

NEOS Server for Optimization

How do we use AMPL once the course is over?

NEOS Server for Optimization

How do we use AMPL once the course is over?

NEOS Server: https://neos-server.org/neos/

NEOS Server Features

- State-of-the-art solvers (AMPL, GAMS, ...)
- Case studies & optimization guide
- It's all free!

Free demo license for AMPL, limited in size

Test Problem Libraries

We have many optimization test problem libraries:

- The CUTEr/st Test Problem Set (SIF) ... in AMPL
  http://orfe.princeton.edu/~rvdb/ampl/nlmodels/
- GAMS World, www.gamsworld.org/
- MacMINLP, My AMPL Collection of MINLPs
  http://wiki.mcs.anl.gov/leyffer/index.php/MacMINLP
- Global Optimization,
  http://titan.princeton.edu/TestProblems

... and many more

Test Problem Libraries

We have many optimization test problem libraries:

- The CUTEr/st Test Problem Set (SIF) ... in AMPL
  http://orfe.princeton.edu/~rvdb/ampl/nlmodels/
- GAMS World, www.gamsworld.org/
- MacMINLP, My AMPL Collection of MINLPs
  http://wiki.mcs.anl.gov/leyffer/index.php/MacMINLP
- Global Optimization,
  http://titan.princeton.edu/TestProblems

... and many more

Test problem libraries are good place to start to learn AMPL!

Course Outline

Part I: Introduction to Optimization

1. Optimization problems, classification, simple methods.
2. Optimization models, algebraic modeling languages.

Part II: Unconstrained Optimization

1. Optimality conditions.
2. Numerical methods & convergence analysis.

Goals

- Preview of key algorithmic ingredients.
- Modeling optimization problems.

Course Outline

Part III: General Nonlinear Optimization problems

1. Optimality conditions.
2. Special problems: linear and quadratic optimization.
3. Methods: local step & global convergence.
4. Optimization problems with equilibrium constraints.

Part IV: Mixed-Integer Nonlinear Optimization

1. Modeling with integer variables.
2. Methods for convex and nonconvex problems
3. Mixed-Integer PDE Constrained Optimization

Goals

- Modern methods for nonlinear optimization.
- Mixed-integer optimization models and techniques.

Course Software

- Course Website: http://wiki.mcs.anl.gov/leyffer/
  - See `Course & Lectures` on right
  - Contains pdf of lecture notes, slides, software, tutorials, solutions
- AMPL modeling language
  - See course website for downloads & book
  - Student version available (up to 300 vars/cons)
- COIN-OR solvers (started by IBM and CMU) http://projects.coin-or.org/CoinBinary
  - See instructions for `svn` and installation
  - Needs C++ compiler, e.g. GNU's g++
  - Needs BLAS, see www.netlib.org/blas/
  - Troubleshoot: run `get.AllThirdParty`
  - Easy Install: Ask Prashant for package!

Course Software

- AMPL mode for emacs, and vim editors http://github.com/dpo/ampl-mode/blob/master/emacs/ampl-mode.el
- Set up a bin directory for binaries & add it to your path:

```
cd ~
mkdir bin
```

- Collect all binaries in bin/ ... or symbolic link to them
- Tell your system where to find the binaries:

```
export PATH=~/bin:\$PATH
```

... now you can call your binaries from anywhere.

Summary and Take-Away Points

- Defined optimization problem & its ingredients:
  1. Objective ... goal that we optimize
  2. Variables ... decision variables
  3. Constraints ... restrictions on choices
- Classified optimization problems by these components
- Introduced optimization software eco-system
  - Brief introduction to AMPL
  - Online resources & open-source software
- Course Outline & Course Software