# Optimization software for medium and large-scale problems

Umamahesh Srinivas

iPAL Group Meeting

December 17, 2010

# Outline

- MATLAB Optimization Toolbox
- Problem types and algorithms
- Optimization settings
- Function handles and GUI
- cvx
- Other optimization tools in MATLAB
- GAMS
- Online resources

# MATLAB Optimization Toolbox

- Widely used algorithms for standard and large-scale optimization
- Constrained and unconstrained problems
- Continuous and discrete variables
- Variety of problems:
  - Linear programming (LP)
  - Quadratic programming (QP)
  - Binary integer programming
  - (General) Nonlinear optimization
  - Multi-objective optimization
- Key features:
  - Find optimal solutions
  - Perform tradeoff analysis
  - Balance multiple design alternatives
  - Support for parallel computing.

# MATLAB Optimization Toolbox

- Widely used algorithms for standard and large-scale optimization
- Constrained and unconstrained problems
- Continuous and discrete variables
- Variety of problems:
  - Linear programming (LP)
  - Quadratic programming (QP)
  - Binary integer programming
  - (General) Nonlinear optimization
  - Multi-objective optimization
- Key features:
  - Find optimal solutions
  - Perform tradeoff analysis
  - Balance multiple design alternatives
  - Support for parallel computing.

# MATLAB Optimization Toolbox

- Widely used algorithms for standard and large-scale optimization
- Constrained and unconstrained problems
- Continuous and discrete variables
- Variety of problems:
  - Linear programming (LP)
  - Quadratic programming (QP)
  - Binary integer programming
  - (General) Nonlinear optimization
  - Multi-objective optimization
- Key features:
  - Find optimal solutions
  - Perform tradeoff analysis
  - Balance multiple design alternatives
  - Support for parallel computing.

# Algorithms for specific types of problems

- Continuous variables
  - Constrained, convex
    - Linear program: `linprog`
    - Quadratic program: `quadprog`
  - Nonlinear
    - Unconstrained: `fminunc` (local minimum of multivariate function), `fminsearch`
    - Constrained: `fmincon`, `fminbnd` (single variable bounded minimization), `fseminf` (semi-infinite constraints)
    - Example of `fseminf`: minimize $(x-1)^2$, subject to $0 \leq x \leq 2$ and $g(x,t) = (x-0.5) - (t-0.5)^2 \leq 0, \ \forall \ t \in [0,1]$.
  - Least squares
    - Linear objective, constrained: `lsqnonneg`, `lsqlin`
    - Nonlinear objective: `lsqnonlin`, `lsqcurvefit`
  - Multi-objective: `fgoalattain`, `fminimax`
- Discrete variables
  - Binary integer programming: `bintprog`
  - Minimize $f(\mathbf{x})$ subject to $\mathbf{Ax} = \mathbf{b}, \mathbf{Cx} \leq \mathbf{d}$
  - $\mathbf{x} \in \{0,1\}^n$.

# Algorithms for specific types of problems

- Continuous variables
  - Constrained, convex
    - Linear program: `linprog`
    - Quadratic program: `quadprog`
  - Nonlinear
    - Unconstrained: `fminunc` (local minimum of multivariate function), `fminsearch`
    - Constrained: `fmincon`, `fminbnd` (single variable bounded minimization), `fseminf` (semi-infinite constraints)
    - Example of `fseminf`: minimize $(x-1)^2$, subject to $0 \leq x \leq 2$ and $g(x,t) = (x-0.5) - (t-0.5)^2 \leq 0,\ \forall\, t \in [0,1]$.
  - Least squares
    - Linear objective, constrained: `lsqnonneg`, `lsqlin`
    - Nonlinear objective: `lsqnonlin`, `lsqcurvefit`
  - Multi-objective: `fgoalattain`, `fminimax`
- Discrete variables
  - Binary integer programming: `bintprog`
  - Minimize $f(\mathbf{x})$ subject to $\mathbf{Ax} = \mathbf{b}, \mathbf{Cx} \leq \mathbf{d}$
  - $\mathbf{x} \in \{0,1\}^n$.

# Algorithms for specific types of problems

- Continuous variables
  - Constrained, convex
    - Linear program: `linprog`
    - Quadratic program: `quadprog`
  - Nonlinear
    - Unconstrained: `fminunc` (local minimum of multivariate function), `fminsearch`
    - Constrained: `fmincon`, `fminbnd` (single variable bounded minimization), `fseminf` (semi-infinite constraints)
    - Example of `fseminf`: minimize $(x-1)^2$, subject to $0 \le x \le 2$ and $g(x,t) = (x-0.5) - (t-0.5)^2 \le 0, \ \forall\, t \in [0,1]$.
  - Least squares
    - Linear objective, constrained: `lsqnonneg`, `lsqlin`
    - Nonlinear objective: `lsqnonlin`, `lsqcurvefit`
  - Multi-objective: `fgoalattain`, `fminimax`
- Discrete variables
  - Binary integer programming: `bintprog`
  - Minimize $f(\mathbf{x})$ subject to $\mathbf{Ax} = \mathbf{b}, \mathbf{Cx} \le \mathbf{d}$
  - $\mathbf{x} \in \{0,1\}^n$.

# Algorithms for specific types of problems

- Continuous variables
  - Constrained, convex
    - Linear program: `linprog`
    - Quadratic program: `quadprog`
  - Nonlinear
    - Unconstrained: `fminunc` (local minimum of multivariate function), `fminsearch`
    - Constrained: `fmincon`, `fminbnd` (single variable bounded minimization), `fseminf` (semi-infinite constraints)
    - Example of `fseminf`: minimize $(x-1)^2$, subject to $0 \leq x \leq 2$ and $g(x,t) = (x-0.5) - (t-0.5)^2 \leq 0, \ \forall \ t \in [0,1]$.
  - Least squares
    - Linear objective, constrained: `lsqnonneg`, `lsqlin`
    - Nonlinear objective: `lsqnonlin`, `lsqcurvefit`
  - Multi-objective: `fgoalattain`, `fminimax`
- Discrete variables
  - Binary integer programming: `bintprog`
  - Minimize $f(\mathbf{x})$ subject to $\mathbf{Ax} = \mathbf{b}, \mathbf{Cx} \leq \mathbf{d}$
  - $\mathbf{x} \in \{0,1\}^n$.

# Algorithms for specific types of problems

- Continuous variables
  - Constrained, convex
    - Linear program: `linprog`
    - Quadratic program: `quadprog`
  - Nonlinear
    - Unconstrained: `fminunc` (local minimum of multivariate function), `fminsearch`
    - Constrained: `fmincon`, `fminbnd` (single variable bounded minimization), `fseminf` (semi-infinite constraints)
    - Example of `fseminf`: minimize $(x-1)^2$, subject to $0 \leq x \leq 2$ and $g(x,t) = (x-0.5) - (t-0.5)^2 \leq 0, \ \forall\ t \in [0,1]$.
  - Least squares
    - Linear objective, constrained: `lsqnonneg`, `lsqlin`
    - Nonlinear objective: `lsqnonlin`, `lsqcurvefit`
  - Multi-objective: `fgoalattain`, `fminimax`
- Discrete variables
  - Binary integer programming: `bintprog`
  - Minimize $f(\mathbf{x})$ subject to $\mathbf{Ax} = \mathbf{b}, \mathbf{Cx} \leq \mathbf{d}$
  - $\mathbf{x} \in \{0,1\}^n$.

# Algorithms for specific types of problems

- Continuous variables
  - Constrained, convex
    - Linear program: `linprog`
    - Quadratic program: `quadprog`
  - Nonlinear
    - Unconstrained: `fminunc` (local minimum of multivariate function), `fminsearch`
    - Constrained: `fmincon`, `fminbnd` (single variable bounded minimization), `fseminf` (semi-infinite constraints)
    - Example of `fseminf`: minimize $(x-1)^2$, subject to $0 \leq x \leq 2$ and $g(x,t) = (x-0.5) - (t-0.5)^2 \leq 0,\ \forall\, t \in [0,1]$.
  - Least squares
    - Linear objective, constrained: `lsqnonneg`, `lsqlin`
    - Nonlinear objective: `lsqnonlin`, `lsqcurvefit`
  - Multi-objective: `fgoalattain`, `fminimax`
- Discrete variables
  - Binary integer programming: `bintprog`
  - Minimize $f(\mathbf{x})$ subject to $\mathbf{Ax} = \mathbf{b}, \mathbf{Cx} \leq \mathbf{d}$
  - $\mathbf{x} \in \{0,1\}^n$.

# Optimization problem syntax

Example LP:

$$\begin{array}{ll} \text{minimize} & \mathbf{c}^T\mathbf{x} \\ \text{subject to} & \mathbf{A}\mathbf{x} \le \mathbf{b} \\ & \mathbf{C}\mathbf{x} = \mathbf{d} \\ & \mathbf{l} \le \mathbf{x} \le \mathbf{u} \end{array}$$

MATLAB syntax:

```
[x,fval,exitflag,output,lambda]
= linprog(c,A,b,C,d,l,u,...
x0,options)
```

- `x0`: initial value for $\mathbf{x}$
- `exitflag`: reason for algorithm termination; useful for debugging
- `lambda`: Lagrangian multipliers
- `output`: Number of iterations, algorithm used, etc.

# Optimization problem syntax

Example LP:

$$
\begin{array}{ll}
\text{minimize} & \mathbf{c}^T\mathbf{x} \\
\text{subject to} & \mathbf{A}\mathbf{x} \leq \mathbf{b} \\
& \mathbf{C}\mathbf{x} = \mathbf{d} \\
& \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}
\end{array}
$$

MATLAB syntax:

```
[x,fval,exitflag,output,lambda]
= linprog(c,A,b,C,d,l,u,...
x0,options)
```

- `x0`: initial value for $\mathbf{x}$
- `exitflag`: reason for algorithm termination; useful for debugging
- `lambda`: Lagrangian multipliers
- `output`: Number of iterations, algorithm used, etc.

# Choosing an algorithm

- Large-scale
  - Uses linear algebra that does not need to store, or operate on, full matrices
  - Preserves sparsity structure
- Medium-scale
  - Internally creates full matrices
  - Requires lot of memory
  - Time-intensive computations.

# Unconstrained nonlinear algorithms

- `fminunc`
  - Large-scale: user-supplied Hessian, or finite difference approximation
  - Medium-scale: cubic line search; uses quasi-Newton updates of Hessian
- `fminsearch`
  - Derivative-free method (Nelder-Mead simplex)
- `fsolve`
  - Trust-region-dogleg: specially designed for nonlinear equations
  - Trust-region reflective: effective for large-scale (sparse) problems
  - Levenberg-Marquardt

# Unconstrained nonlinear algorithms

- `fminunc`
  - Large-scale: user-supplied Hessian, or finite difference approximation
  - Medium-scale: cubic line search; uses quasi-Newton updates of Hessian
- `fminsearch`
  - Derivative-free method (Nelder-Mead simplex)
- `fsolve`
  - Trust-region-dogleg: specially designed for nonlinear equations
  - Trust-region reflective: effective for large-scale (sparse) problems
  - Levenberg-Marquardt

# Unconstrained nonlinear algorithms

- `fminunc`
  - Large-scale: user-supplied Hessian, or finite difference approximation
  - Medium-scale: cubic line search; uses quasi-Newton updates of Hessian
- `fminsearch`
  - Derivative-free method (Nelder-Mead simplex)
- `fsolve`
  - Trust-region-dogleg: specially designed for nonlinear equations
  - Trust-region reflective: effective for large-scale (sparse) problems
  - Levenberg-Marquardt

# Constrained linear and nonlinear algorithms

- `fmincon`
  - Trust-region reflective: subspace trust-region method; user-specified gradient; special techniques like Hessian multiply; large-scale
  - Active set: sequential quadratic programming method; medium-scale method; large step size (faster)
  - Interior point: log-barrier penalty term for inequality constraints; problem reduced to having only equality constraints; large-scale
- `linprog`
  - Large-scale interior point
  - Medium-scale active set
  - Medium-scale simplex
- `quadprog`, `lsqlin`
  - Large-scale
  - Medium-scale
- `lsqcurvefit`, `lsqnonlin`
  - Trust-region reflective: effective for large-scale (sparse) problems
  - Levenberg-Marquardt

# Constrained linear and nonlinear algorithms

- `fmincon`
  - Trust-region reflective: subspace trust-region method; user-specified gradient; special techniques like Hessian multiply; large-scale
  - Active set: sequential quadratic programming method; medium-scale method; large step size (faster)
  - Interior point: log-barrier penalty term for inequality constraints; problem reduced to having only equality constraints; large-scale
- `linprog`
  - Large-scale interior point
  - Medium-scale active set
  - Medium-scale simplex
- `quadprog`, `lsqlin`
  - Large-scale
  - Medium-scale
- `lsqcurvefit`, `lsqnonlin`
  - Trust-region reflective: effective for large-scale (sparse) problems
  - Levenberg-Marquardt

# Constrained linear and nonlinear algorithms

- `fmincon`
  - Trust-region reflective: subspace trust-region method; user-specified gradient; special techniques like Hessian multiply; large-scale
  - Active set: sequential quadratic programming method; medium-scale method; large step size (faster)
  - Interior point: log-barrier penalty term for inequality constraints; problem reduced to having only equality constraints; large-scale
- `linprog`
  - Large-scale interior point
  - Medium-scale active set
  - Medium-scale simplex
- `quadprog`, `lsqlin`
  - Large-scale
  - Medium-scale
- `lsqcurvefit`, `lsqnonlin`
  - Trust-region reflective: effective for large-scale (sparse) problems
  - Levenberg-Marquardt

# Constrained linear and nonlinear algorithms

- `fmincon`
  - Trust-region reflective: subspace trust-region method; user-specified gradient; special techniques like Hessian multiply; large-scale
  - Active set: sequential quadratic programming method; medium-scale method; large step size (faster)
  - Interior point: log-barrier penalty term for inequality constraints; problem reduced to having only equality constraints; large-scale
- `linprog`
  - Large-scale interior point
  - Medium-scale active set
  - Medium-scale simplex
- `quadprog`, `lsqlin`
  - Large-scale
  - Medium-scale
- `lsqcurvefit`, `lsqnonlin`
  - Trust-region reflective: effective for large-scale (sparse) problems
  - Levenberg-Marquardt

# `linprog` demo

A farmer wants to decide how to grow two crops $x$ and $y$ on 75 acres of land in order to maximize his profit $143x + 60y$, subject to a storage constraint $110x + 30y \leq 4000$ and an investment constraint $120x + 210y \leq 15000$.

$$\begin{array}{ll} \text{minimize} & -(143x + 60y) \\ \text{subject to} & x + y \leq 75 \\ & 110x + 30y \leq 4000 \\ & 120x + 210y \leq 15000 \\ & -x \leq 0 \\ & -y \leq 0. \end{array}$$

# quadprog demo 1

Support vector machine (SVM):
Given the set of labeled points $\{\mathbf{x}_i, y_i\}_{i=1}^m$, $y_i \in \{-1, +1\}$, which is linearly separable, find the vector $\mathbf{w}$ which defines the hyperplane separating the set with maximum margin, i.e.,

$$\begin{aligned} \mathbf{x}_i^T\mathbf{w} - b &\geq 1, \text{ if } y_i = 1 \\ \mathbf{x}_i^T\mathbf{w} - b &\leq -1, \text{ if } y_i = -1. \end{aligned}$$

$$\mathbf{A} := \begin{bmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_m^T \end{bmatrix}, \mathbf{D} = \mathrm{diag}(y_1, \ldots, y_m) \Rightarrow \mathbf{D}(\mathbf{A}\mathbf{w} - \mathbf{b}) \geq \mathbf{1}$$

$$\begin{aligned} \text{minimize} \quad & \|\mathbf{w}\|_2^2 \\ \text{subject to} \quad & \mathbf{D}(\mathbf{A}\mathbf{w} - \mathbf{b}) \geq \mathbf{1}. \end{aligned}$$

# quadprog demo 1

Support vector machine (SVM):
Given the set of labeled points $\{\mathbf{x}_i, y_i\}_{i=1}^m$, $y_i \in \{-1, +1\}$, which is linearly separable, find the vector $\mathbf{w}$ which defines the hyperplane separating the set with maximum margin, i.e.,

$$\begin{aligned} \mathbf{x}_i^T\mathbf{w} - b &\geq 1, \text{ if } y_i = 1 \\ \mathbf{x}_i^T\mathbf{w} - b &\leq -1, \text{ if } y_i = -1. \end{aligned}$$

$$\mathbf{A} := \begin{bmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_m^T \end{bmatrix}, \mathbf{D} = \mathrm{diag}(y_1, \ldots, y_m) \Rightarrow \mathbf{D}(\mathbf{A}\mathbf{w} - \mathbf{b}) \geq \mathbf{1}$$

$$\begin{array}{ll} \text{minimize} & \|\mathbf{w}\|_2^2 \\ \text{subject to} & \mathbf{D}(\mathbf{A}\mathbf{w} - \mathbf{b}) \geq \mathbf{1}. \end{array}$$

# quadprog demo 2

$$\begin{array}{ll} \text{minimize} & 4x_1^2 + x_1x_2 + 4x_2^2 + 3x_1 - 4x_2 \\ \text{subject to} & x_1 + x_2 \leq 5 \\ & -x_1 \leq 0 \\ & -x_2 \leq 0 \\ & x_1 - x_2 = 0. \end{array}$$

$$f(x_1, x_2) = \tfrac{1}{2}[x_1 \ \ x_2]^T \begin{bmatrix} 8 & 1 \\ 1 & 8 \end{bmatrix} [x_1 \ \ x_2] + [3 \ \ -4]^T [x_1 \ \ x_2].$$

# Function handles and GUI

- A standard MATLAB data type that provides a means of calling a function indirectly: `handle = @functionname`
- Widely used in optimization
- MATLAB GUI for optimization: `optimtool`.

# fmincon demo

Rosenbrock function:

$$\begin{array}{ll} \text{minimize} & 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \\ \text{subject to} & x_1^2 + x_2^2 \leq 1. \end{array}$$

Choose $[0,\ 0]^T$ as the initial point.

# cvx[1]

- MATLAB-based modeling system for *disciplined convex programming*
- Supports the formulation and construction of convex optimization problems (convexity to be ensured by user)
- Features:
  - SeDuMi and SDPT3 interior-point solvers
  - Well-defined problems (LP, SOCP, SDP) handled exactly
  - Supports functions that are (good) approximations of convex problems, or can be obtained from successive convex approximations.

[1] Developed by Michael Grant and Stephen Boyd, Stanford

# cvx demo

Quadratic program:

$$
\begin{array}{ll}
\text{minimize} & 4x_1^2 + x_1x_2 + 4x_2^2 + 3x_1 - 4x_2 \\
\text{subject to} & x_1 + x_2 \leq 5 \\
& -x_1 \leq 0 \\
& -x_2 \leq 0 \\
& x_1 - x_2 = 0.
\end{array}
$$

$$f(x_1, x_2) = \frac{1}{2}[x_1 \;\; x_2]^T \begin{bmatrix} 8 & 1 \\ 1 & 8 \end{bmatrix} [x_1 \;\; x_2] + [3 \;\; -4]^T [x_1 \;\; x_2].$$

# Other optimization tools in MATLAB

- Genetic algorithms and direct search toolbox
  - Derivative-free methods
  - Pattern search
  - Simulated annealing
- Global optimization toolbox
  - Global solutions to problems with multiple extrema
  - General objective and constraint functions:
    - continuous/discontinuous
    - stochastic
    - may include simulations or black-box functions
    - may not possess derivatives.

# Other optimization tools in MATLAB

- Genetic algorithms and direct search toolbox
  - Derivative-free methods
  - Pattern search
  - Simulated annealing
- Global optimization toolbox
  - Global solutions to problems with multiple extrema
  - General objective and constraint functions:
    - continuous/discontinuous
    - stochastic
    - may include simulations or black-box functions
    - may not possess derivatives.

# TOMLAB

- General-purpose development and modeling environment for optimization problems
- Compatible with MATLAB Optimization Toolbox
- MATLAB solver algorithms, as well as state-of-the-art optimization software packages

Key features:

- Faster and more robust compared to built-in MATLAB solvers
- Automatic, efficient differentiation
- Robust solution of ill-conditioned nonlinear least squares problems with linear constraints using several different solver options
- Special treatment of exponential fitting and other types of nonlinear parameter estimation problems.

# TOMLAB

- General-purpose development and modeling environment for optimization problems
- Compatible with MATLAB Optimization Toolbox
- MATLAB solver algorithms, as well as state-of-the-art optimization software packages

Key features:

- Faster and more robust compared to built-in MATLAB solvers
- Automatic, efficient differentiation
- Robust solution of ill-conditioned nonlinear least squares problems with linear constraints using several different solver options
- Special treatment of exponential fitting and other types of nonlinear parameter estimation problems.

# GAMS

- General Algebraic Modeling System
- User-friendly syntax
- Suitable for large-scale problems
- Different types of solvers:
    - Linear programs
    - Mixed integer programs
    - Nonlinear programs
    - Constrained nonlinear systems
    - Quadratically constrained programs
    - Mixed complementarity problems
- Facilitates sensitivity analysis

# Illustrative example: Linear program

- Indices: $i =$ plants, $j =$ markets
- Given data:
  - $a_i =$ supply of commodity at plant $i$
  - $b_j =$ demand for commodity at market $j$
  - $c_{ij} =$ cost per unit shipment between plant $i$ and market $j$
- Decision variable: $x_{ij} =$ amount of commodity to ship from plant $i$ to market $j$
- Constraints:
  - $x_{ij} \geq 0$
  - Observe supply limit at plant $i$: $\sum_j x_{ij} \leq a_i \ \forall \ i$
  - Satisfy demand at market $j$: $\sum_i x_{ij} \geq b_j \ \forall \ j$
- Objective function: minimize $\sum_{i,j} c_{ij} x_{ij}$

# Modeling in GAMS

Good modeling practices:

- Model entities identified and grouped by type
- No symbol referred to before it is defined

Terminology:

- Indices $\rightarrow$ `sets`
- Given data $\rightarrow$ `parameters`
- Decision variables $\rightarrow$ `variables`
- Constraints and objective functions $\rightarrow$ `equations`

# GAMS demo

| | Shipping distance ($\times 1000$ miles) | | | Supplies |
|---|---|---|---|---|
| | Markets | | | |
| Plants | New York | Chicago | Topeka | |
| Seattle | 2.5 | 1.7 | 1.8 | 350 |
| San Diego | 2.5 | 1.8 | 1.4 | 600 |
| Demand | 325 | 300 | 275 | |

Unit shipping cost: $90

# Structure of a GAMS model

| Inputs | Outputs |
|---|---|
| `Sets` | Echo print |
| Declaration | Reference maps |
| Assignment of members | Equation listings |
| | Status reports |
| Data (`Parameters, Tables, Scalar`) | Results |
| Declaration, assignment of values | |
| `Variables` | |
| Declaration - assignment of type | |
| Bounds, initial values (optional) | |
| `Equations` | |
| Declaration, definition | |
| `Model` and `Solve` | |

# GAMS: Advantages

- Algebra-based notation: easy-to-read for both user and computer
- Reusability of GAMS models
- Model documentation
- Output report easy to interpret
- Extensive debugging ability of compiler
- Models scalable for large problems

# Penn State computing resources

- High Performance Computing (HPC) group
- Develops and maintains state-of-the-art computational clusters
- Separate clusters for interactive and batch programming
- Support and expertise for research using programming languages, numerical libraries, statistical packages, finite element solvers, and specialized software
- Expertise for code optimization and parallelization on high performance computing machines
- University-wide access to variety of licensed computational software
- Seminars on data-intensive and numerically-intensive computing

# Penn State computing resources

- High Performance Computing (HPC) group
- Develops and maintains state-of-the-art computational clusters
- Separate clusters for interactive and batch programming
- Support and expertise for research using programming languages, numerical libraries, statistical packages, finite element solvers, and specialized software
- Expertise for code optimization and parallelization on high performance computing machines
- University-wide access to variety of licensed computational software
- Seminars on data-intensive and numerically-intensive computing

# Penn State computing resources

- High Performance Computing (HPC) group
- Develops and maintains state-of-the-art computational clusters
- Separate clusters for interactive and batch programming
- Support and expertise for research using programming languages, numerical libraries, statistical packages, finite element solvers, and specialized software
- Expertise for code optimization and parallelization on high performance computing machines
- University-wide access to variety of licensed computational software
- Seminars on data-intensive and numerically-intensive computing

# Online resources

- MATLAB Optimization Toolbox documentation
- Detailed listing of `optimset` options
- cvx user guide
- NEOS Optimization software guide
- TOMLAB documentation
- GAMS documentation