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 \(f(x)\) subject to \(Ax = b, Cx \leq d\)
  - \(x \in \{0, 1\}^n\).
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 \(f(x)\) subject to \(Ax = b, Cx \leq d\)
  - \(x \in \{0, 1\}^n\).
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 \(f(x)\) subject to \(Ax = b, Cx \leq d\)
  - \(x \in \{0, 1\}^n\).
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`, `lsqref`
  - **Nonlinear objective:** `lsqnonlin`, `lsqcurvefit`
  - **Multi-objective:** `fgoalattain`, `fminimax`

- **Discrete variables**
  - Binary integer programming: `bintprog`
  - Minimize \(f(x)\) subject to \(Ax = b, Cx \leq d\)
  - \(x \in \{0, 1\}^n\).
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 $f(x)$ subject to $Ax = b$, $Cx \leq d$
  - $x \in \{0, 1\}^n$.  

12/17/2010  iPAL Group Meeting
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 
      \[g(x, t) = (x - 0.5) - (t - 0.5)^2 \leq 0, \quad \forall \ t \in [0, 1].\]

- **Least squares**
  - Linear objective, constrained: lsqnonneg, lsqlin
  - Nonlinear objective: lsqnonlin, lsqcurvefit

- **Multi-objective**: fgoalattain, fminimax

- **Discrete variables**
  - Binary integer programming: bintprog
  - Minimize \(f(x)\) subject to \(Ax = b, Cx \leq d\)
  - \(x \in \{0, 1\}^n\).
Optimization problem syntax

Example LP:

\[
\begin{align*}
\text{minimize} & \quad c^T x \\
\text{subject to} & \quad Ax \leq b \\
& \quad Cx = d \\
& \quad l \leq x \leq u
\end{align*}
\]

MATLAB syntax:

\[
[x,fval,exitflag,output,lambda] = \text{linprog}(c,A,b,C,d,l,u,\ldots)
\]

- \(x0\): initial value for \(x\)
- \(exitflag\): reason for algorithm termination; useful for debugging
- \(\lambda\): Lagrangian multipliers
- \(\text{output}\): Number of iterations, algorithm used, etc.
Optimization problem syntax

Example LP:

\[
\begin{align*}
\text{minimize} & \quad c^T x \\
\text{subject to} & \quad Ax \leq b \\
& \quad Cx = d \\
& \quad l \leq x \leq u
\end{align*}
\]

MATLAB syntax:

\[
[x,fval,exitflag,output,lambda] = \text{linprog}(c,A,b,C,d,l,u,\ldots)
\]

- \(x0\): initial value for \(x\)
- \(exitflag\): 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

12/17/2010 iPAL Group Meeting
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
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 $120x + 210y \leq 15000$.

\[
\begin{align*}
\text{minimize} & \quad -(143x + 60y) \\
\text{subject to} & \quad x + y \leq 75 \\
& \quad 110x + 30y \leq 4000 \\
& \quad 120x + 210y \leq 15000 \\
& \quad -x \leq 0 \\
& \quad -y \leq 0.
\end{align*}
\]
Support vector machine (SVM):
Given the set of labeled points \( \{x_i, y_i\}_{i=1}^m \), \( y_i \in \{-1, +1\} \), which is linearly separable, find the vector \( w \) which defines the hyperplane separating the set with maximum margin, i.e.,

\[
\begin{align*}
  x_i^T w - b & \geq 1, \text{ if } y_i = 1 \\
  x_i^T w - b & \leq -1, \text{ if } y_i = -1.
\end{align*}
\]

\[
A := \begin{bmatrix}
x_1^T \\
\vdots \\
x_m^T
\end{bmatrix}, \quad D = \text{diag}(y_1, \ldots, y_m) \Rightarrow D(Aw - b) \geq 1
\]

minimize \( \|w\|_2^2 \)
subject to \( D(Aw - b) \geq 1. \)
Support vector machine (SVM):
Given the set of labeled points \( \{x_i, y_i\}_{i=1}^m, y_i \in \{-1, +1\} \), which is linearly separable, find the vector \( w \) which defines the hyperplane separating the set with maximum margin, i.e.,

\[
\begin{align*}
    x_i^T w - b & \geq 1, \text{ if } y_i = 1 \\
    x_i^T w - b & \leq -1, \text{ if } y_i = -1.
\end{align*}
\]

\[
A := \begin{bmatrix}
    x_1^T \\
    \vdots \\
    x_m^T
\end{bmatrix}, \quad D = \text{diag}(y_1, \ldots, y_m) \Rightarrow D(Aw - b) \geq 1
\]

minimize \( \|w\|_2^2 \)
subject to \( D(Aw - b) \geq 1 \).
quadprog demo 2

minimize \[ 4x_1^2 + x_1x_2 + 4x_2^2 + 3x_1 - 4x_2 \]
subject to \[ x_1 + x_2 \leq 5 \]
\[ -x_1 \leq 0 \]
\[ -x_2 \leq 0 \]
\[ x_1 - x_2 = 0. \]

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

\[
\begin{align*}
\text{minimize} & \quad 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \\
\text{subject to} & \quad x_1^2 + x_2^2 \leq 1.
\end{align*}
\]

Choose \([0, 0]^T\) as the initial point.
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
Quadratic program:

\[
\begin{align*}
\text{minimize} & \quad 4x_1^2 + x_1x_2 + 4x_2^2 + 3x_1 - 4x_2 \\
\text{subject to} & \quad x_1 + x_2 \leq 5 \\
& \quad -x_1 \leq 0 \\
& \quad -x_2 \leq 0 \\
& \quad x_1 - x_2 = 0.
\end{align*}
\]

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

12/17/2010 iPAL Group Meeting
Modeling in GAMS

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

Terminology:
- Indices → sets
- Given data → parameters
- Decision variables → variables
- Constraints and objective functions → equations
### Shipping distance (×1000 miles) and Supplies

<table>
<thead>
<tr>
<th>Plants</th>
<th>New York</th>
<th>Chicago</th>
<th>Topeka</th>
<th>Supplies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seattle</td>
<td>2.5</td>
<td>1.7</td>
<td>1.8</td>
<td>350</td>
</tr>
<tr>
<td>San Diego</td>
<td>2.5</td>
<td>1.8</td>
<td>1.4</td>
<td>600</td>
</tr>
<tr>
<td>Demand</td>
<td>325</td>
<td>300</td>
<td>275</td>
<td></td>
</tr>
</tbody>
</table>

Unit shipping cost: $90
Structure of a GAMS model

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sets</td>
<td>Echo print</td>
</tr>
<tr>
<td>Declaration</td>
<td>Reference maps</td>
</tr>
<tr>
<td>Assignment of members</td>
<td>Equation listings</td>
</tr>
<tr>
<td>Data (Parameters, Tables, Scalar)</td>
<td>Status reports</td>
</tr>
<tr>
<td>Declaration, assignment of values</td>
<td>Results</td>
</tr>
<tr>
<td>Variables</td>
<td></td>
</tr>
<tr>
<td>Declaration - assignment of type</td>
<td></td>
</tr>
<tr>
<td>Bounds, initial values (optional)</td>
<td></td>
</tr>
<tr>
<td>Equations</td>
<td></td>
</tr>
<tr>
<td>Declaration, definition</td>
<td></td>
</tr>
<tr>
<td>Model and Solve</td>
<td></td>
</tr>
</tbody>
</table>
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