Optimieren mit MATLAB – jetzt auch gemischt-ganzzahlig

Dr. Maka Karalashvili
Application Engineer
MathWorks
Let’s consider the following modeling case study...

<table>
<thead>
<tr>
<th>Item</th>
<th>Nuts</th>
<th>Bolts</th>
<th>Revenue per Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gadget</td>
<td>5</td>
<td>2</td>
<td>$3.00</td>
</tr>
<tr>
<td>Widget</td>
<td>3</td>
<td>8</td>
<td>$10.00</td>
</tr>
</tbody>
</table>

Current Inventory

<table>
<thead>
<tr>
<th>Nuts</th>
<th>Bolts</th>
</tr>
</thead>
<tbody>
<tr>
<td>29</td>
<td>34</td>
</tr>
</tbody>
</table>

? How many Gadget / Widget to produce to maximize revenue
How many 🔄 / 🗑 at maximum revenue?

- A non-integer solution
  - 3.8235
  - 3.2941

  - Could attempt to round up/down

- The integer solution

$\begin{array}{l}
\text{not possible} \\
\hline
\text{3.8235} \\
\text{3.2941} \\
\end{array}$

$\begin{array}{l}
\text{not possible} \\
\hline
\text{$44.11$} \\
\text{$42.00$} \\
\text{$43.00$} \\
\end{array}$
Applications of Optimization

For example

- **Portfolio Management**
  - Maximize profits
  - Minimize costs
  - Maximize efficiency
  - Minimize risk

- **Power Generation**

- **Motor Calibration**

- **Manufacturing / Supply Chain**
Optimization Problem
In Literature and in MATLAB

Objective Function

\[ \min_x f(x) \]

Typically a linear or nonlinear function

Decision variables (can be discrete or integer)

Subject to Constraints

**Linear constraints**
- inequalities
- equalities
- bounds

\[ Ax \leq b \]
\[ A_{eq} x = b_{eq} \]
\[ l \leq x \leq u \]

**Nonlinear constraints**
- inequalities
- equalities

\[ c(x) \leq 0 \]
\[ c_{eq}(x) = 0 \]
Approaches in MATLAB

- **Local Optimization**  
  - Optimization Toolbox
  - Finds local minima/maxima
  - Needs supplying gradients
  - Applicable for large scale problems with smooth objective function
  - Faster/fewer function evaluations

- **Global Optimization**  
  - Global Optimization Toolbox
  - In general, no gradient information required
  - Solve problems with non-smooth, stochastic, discontinuous objective function
### Some Types of Optimization Problems

<table>
<thead>
<tr>
<th>Constraint Type</th>
<th>Objective Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>Linear</td>
</tr>
<tr>
<td>Quadratic</td>
<td>Least Squares (LSQ)</td>
</tr>
<tr>
<td>Linear and/or Bounds</td>
<td>Linear Programm (LP)</td>
</tr>
<tr>
<td></td>
<td>Quadratic Programm (QP)</td>
</tr>
<tr>
<td></td>
<td>Linear/Nonlinear Least Squares (LSQ)</td>
</tr>
</tbody>
</table>

*When discrete or integer values involved*

<table>
<thead>
<tr>
<th>Constraint Type</th>
<th>Objective Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>Linear Mixed-Integer Programm (MILP)</td>
</tr>
</tbody>
</table>

* - nonlinear minimization
Local Solvers in MATLAB

- Linear or quadratic programming problems
  - `linprog`: LP problems
  - `intlinprog`: MILP problems
  - `quadprog`: QP problems

- Least squares problems
  - `lsqlin`: Linear LSQs subject to linear/bound constraints
  - `lsqnonlin`: Nonlinear LSQs subject to bound constraints

- Nonlinear minimization
  - `fminunc`: Unconstrained
  - `fmincon`: Linear and nonlinear constraints
    - Supply Gradient and Hessian functions to speed up

Optimization Toolbox

Choosing the Algorithm
- `fmincon` Algorithms
- `fsolve` Algorithms
- `fminunc` Algorithms
- Least Squares Algorithms
- Linear Programming Algorithms
- Quadratic Programming Algorithms
- Large-Scale vs. Medium-Scale Algorithms
- Potential Inaccuracy with Interior-Point Algorithms
Global Solvers in MATLAB

- **Smooth objective and constraints**
  - `GlobalSearch`, `MultiStart`: Multiple local solutions
  - `GlobalSearch`, `MultiStart`, `patternsearch`, `ga`, `simulannealbnd`: Single global solution

- **Nonsmooth objective or constraints**
  - `patternsearch`, `ga`, `simulannealbnd`: Does not rely on gradient calculation

- **Discrete or integer values**
  - `ga`: Mixed-Integer Nonlinear Problems (MINLP)
    - Can have any objective function, bounds, and inequality constraints
    - Can indirectly include equality constraints
Optimization APP
Optimization APP

Problem Setup and Results

Solver: fmincon - Constrained nonlinear minimization
Algorithm: SQP
Problem: interior point
Objective: SQP
Active set
Derivatives: Trust region reflective

Unboundedness threshold: Use default -1e0
Function value check
Error if user-supplied function returns Inf, NaN or complex

User-supplied derivatives
Validate user-supplied derivatives
Hessian sparsity pattern: Use default: sparse(ones(numberOfVariables))
Hessian multiply function: Use default: No multiply function
Approximated derivatives
Finite differences f(x + h) - f(x)
Type: forward differences
Relative perturbation vector: Use default: sqrt(e)*ones(numberOfVariables,1)
Minimum perturbation |a|: Use default: 0
Maximum perturbation |c|: Use default: Inf
Evaluate in parallel
Optimization APP

![Optimization Tool](image-url)

- **Solver**: `fmincon - Constrained nonlinear minimization`
- **Algorithm**: SQP
- **Objective function**: `@myObj`
- **Start point**: `x0`
- **Derivatives**: `Approximated by solver`
- **Bounds**: Linear inequalities and linear equalities
- **Nonlinear constraint function**: `@myNonCon`
- **Options**: SQP constraint tolerance: Use default 1e-6
- **Run solver and view results**: Start, Pause, Stop
Optimization APP
Optimization APP
Optimization APP

Generate Code...

% This is an auto generated MATLAB file from Optimization Tool.

% Start with the default options
options = optimoptions('fmincon');

% Modify options setting
options = optimoptions(options,'Display', 'off');
options = optimoptions(options,'PlotFcns', { @optimplotx @optimplotfval });
options = optimoptions(options,'Algorithm', 'active-set');
[x,fval,exitflag,output,lambda,grad,hessian] = ...
   fmincon(@rosenbrock,x0,[],[],[],[],[],[],[],options);
Speed up Optimization using Built-in Parallel Support

Iteration n:  
1. Evaluate objective function at $X_n = [x_n, y_n]$  
2. Calculate gradient by finite difference approximation

$$f'(x_n) = \frac{f(x_n + \Delta x_n) - f(x_n)}{\Delta x_n}$$

3. Move to

$$x_{n+1} = x_n + \alpha \ f'(x_n)$$

→ Can run in parallel!
Speed up Optimization using Built-in Parallel Support

1) Gradient Estimation
   - `fmincon`
   - `fminimax`
   - `fgoalattain`

2) Iterative sampling of local solution space
   - `MultiStart`
   - `ga, gamultiobj`
   - `patternsearch`
Key takeaways

1. Solve wide variety of problems
   - Linear, linear mixed-integer, quadratic, nonlinear, least squares
   - Nonlinear, nonsmooth, stochastic, nonlinear mixed-integer

2. MATLAB environment
   - User-friendly graphical interfaces – Apps – with automatic code generation
     - Optimization App
   - Integrated Numeric, Graphics, Symbolic Math
   - Parallel computing

3. Deploying Applications with MATLAB
   - Share applications with end users who do not need MATLAB
     - Stand-alone executables (.exe)
     - Shared libraries (.dll)
     - Software components (.jar, .dll, .com)
Optimization Solvers in MATLAB

**Linear**
- linprog
- intlinprog

**Quadratic**
- quadprog

**Nonlinear**
- fmincon
- multistart
- globalsearch
- patternsearch

**Least Squares**
- lsqlin
- lsqnonneg
- lsqcurvefit
- lsqnonlin

**Nonsmooth or Noisy**
- patternsearch
- ga
- simulannealbnd

**Multiobjective**
- gamultiobj
Backup
Deploy your optimization application

1.) Define the user interface
2.) Package the application using MATLAB Compiler
3.) Give the application installer to someone

They will install the application ... and run it on their desktop