Optimization in Matlab

Kevin Carlberg

Stanford University

July 28, 2009

Overview

Matlab has two toolboxes that contain optimization algorithms discussed in this class

- Optimization Toolbox
  - Unconstrained nonlinear
  - Constrained nonlinear
  - Simple convex: LP, QP
  - Least Squares
  - Binary Integer Programming
  - Multiobjective
- Genetic Algorithm and Direct Search Toolbox: general optimization problems
  - Direct search algorithms (directional): generalized pattern search and mesh adaptive search
  - Genetic algorithm
  - Simulated annealing and Threshold acceptance

Problem types and algorithms

- Continuous
  - Convex, constrained (simple)
    - LP: `linprog`
    - QP: `quadprog`
  - Nonlinear
    - Unconstrained: `fminunc`, `fminsearch`
    - Constrained: `fmincon`, `fminbnd`, `fseminf`
  - Least-squares (specialized problem type): $\min_x \|F(x)\|_2$
    - $F(x)$ linear, constrained: `lsqnonneg`, `lsqlin`
    - $F(x)$ nonlinear: `lsqnonlin`, `lsqcurvefit`
  - Multiobjective: `fgoalattain`, `fminimax`
- Discrete
  - Linear, Binary Integer Programming: `bintprog`

Continuous, nonlinear algorithms

- We will focus only on the following algorithms for continuous, nonlinear problems
- Unconstrained: `fminunc`, `fminsearch`
- Constrained: `fmincon`

Nonlinear, unconstrained algorithms

- `fminunc`: a gradient-based algorithm with two modes
  - **Large-scale**: This is a subspace trust-region method (see p. 76–77 of Nocedal and Wright). It can take a user-supplied Hessian or approximate it using finite differences (with a specified sparsity pattern)
  - **Medium-scale**: This is a cubic line-search method. It uses Quasi-Newton updates of the Hessian (recall that Quasi-Newton updates give dense matrices, which are impractical for large-scale problems)
- `fminsearch`: a derivative-free method based on Nelder-Mead simplex

Nonlinear constrained algorithm: fmincon

- `fmincon`: a gradient-based framework with three algorithms
  - **Trust-region reflective**: a subspace trust-region method
  - **Active Set**: a sequential quadratic programming (SQP) method. The Hessian of the Lagrangian is updated using BFGS.
  - **Interior Point**: a log-barrier penalty term is used for the inequality constraints, and the problem is reduced to having only equality constraints

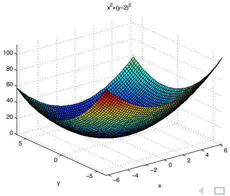
Algorithms

- Algorithms in this toolbox can be used to solve general problems
- All algorithms are derivative-free methods
- Direct search: `patternsearch`
- Genetic algorithm: `ga`
- Simulated annealing/threshold acceptance: `simulannealbnd`, `threshacceptbnd`
- Genetic Algorithm for multiobjective optimization: `gamultiobj`

Function handles

- **Function handle**: a MATLAB value that provides a means of calling a function indirectly
  - Function handles can be passed in calls to other functions
  - Function handles can be stored in data structures for later use
  - The optimization and genetic algorithm toolboxes make extensive use of function handles
- Example: Creating a handle to an anonymous function

```
bowl = @(x,y)x^2+(y-2)^2;
ezsurf(bowl)
```



- Example: creating a handle to a named function
  - At the command line, type

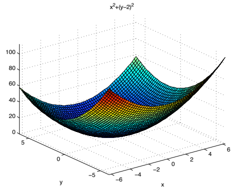
    ```
    edit bowlNamed;
    ```

  - In an editor, create an m-file containing

    ```
    function f = bowlNamed(x,y)
    f = x^2+(y-2)^2;
    ```

  - At the command line, type

    ```
    bowlhandle = @bowlNamed;
    ezsurf(bowlhandle)
    ```



Function handles for optimization

- For the optimization toolbox, only one vector-valued input argument should be used
- Example: creating a handle to an anonymous function with one vector-valued input variable

```
bowlVec = @(x)x(1)^2+(x(2)-2)^2;
```

- Example: creating a handle to a named function with two scalar-valued input variables

```
bowlVecNamed = @(x)bowlNamed(x(1),x(2));
```

- `ezsurf` cannot accept handles with vector-valued arguments (stick with examples on previous pages)

Supplying gradients

- It may be desirable to analytically specify the gradient of the function
- To do this, the named function must return two outputs: the function value *and* the gradient

- Complete example: creating a handle to a named function, plotting it, and specifying the handle for optimization
  - At the command line, type

    ```
    edit bowlNamed;
    ```

  - In the editor, create an m-file containing

    ```
    function [f,g] = bowlNamed(x,y)
    f = x^2+(y-2)^2;
    g(1) = 2*x;
    g(2) = 2*(y-2);
    ```

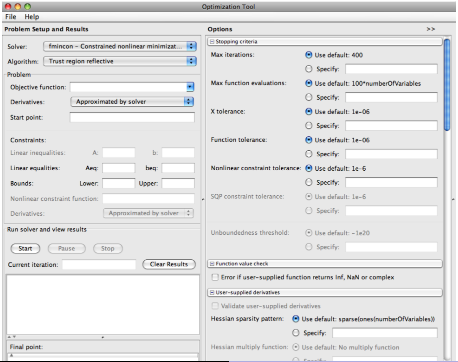
  - At the command line, type

    ```
    bowlhandle = @bowlNamed;
    ezsurf(bowlhandle);
    bowlhandleOpt = @(x) bowlNamed(x(1),x(2))
    ```

- `bowlhandleOpt` can now be used as the argument to a Matlab optimization function with supplied gradients

GUI

- The optimization toolbox includes a graphical user interface (GUI) that is easy to use
- To activate, simply type `optimtool` at the command line



GUI options

- We would like to "track" the progress of the optimizer
- Under `options`, set Level of display: iterative
- Under `plot functions`, check: function value
- When ga is used, check "Best fitness," and "Expectation" to track the fitness of the best member of the population and the average fitness of the population

For the functions on the following pages, do the following:

1. Create a function that takes in two scalar-valued arguments and outputs both the function and gradient
2. Create a handle for this function and use `ezsurf` to plot the function
3. Create an optimization-ready handle for this function and solve using different starting points using:
   - `fminunc`, medium scale, derivatives approximated by solver
   - `fminunc`, medium scale, gradient supplied
   - `fminsearch`
   - `ga`
4. Compare the algorithms on the following measures:
   1. Robustness: ability to find a global optimum and dependence of performance on initial guess
   2. Efficiency: how many function evaluations were required?

Problem 1

- Consider a convex function with constant Hessian

$$f(x_1, x_2) = 4x_1^2 + 7(x_2 - 4)^2 - 4x_1 + 3x_2$$

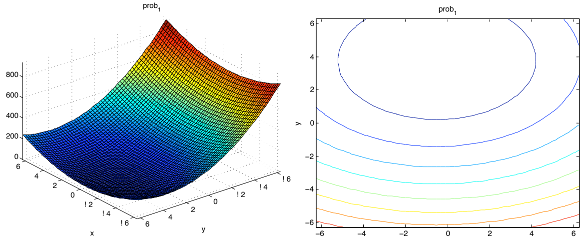


Figure: Surface and contour plot

- Also, *find the analytical solution to this problem*

Problem 2

- Consider the Rosenbrock function, a non-convex problem that is difficult to minimize. The global minimum is located at $(x_1, x_2) = (0, 0)$

$$f(x_1, x_2) = (1 - x_1)^2 + 100(x_2 - x_1^2)^2$$

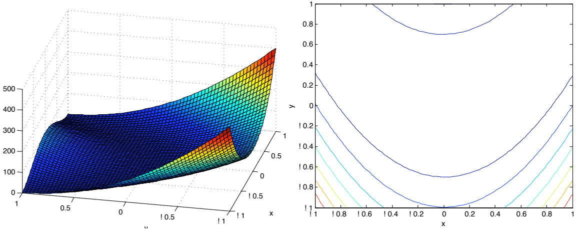


Figure: Surface and contour plot

Problem 3

- Consider the Rastragin's function, an all-around nasty function. The global minimum is located at $(x_1, x_2) = (0, 0)$

$$f(x_1, x_2) = 20 + x_1^2 + x_2^2 - 10(\cos 2\pi x_1 + \cos 2\pi x_2)$$

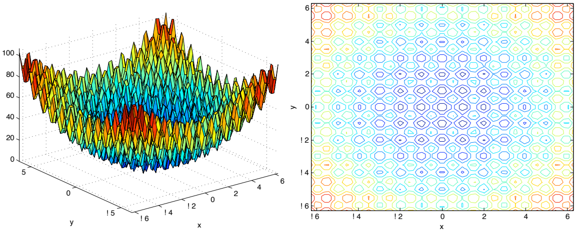


Figure: Surface and contour plot