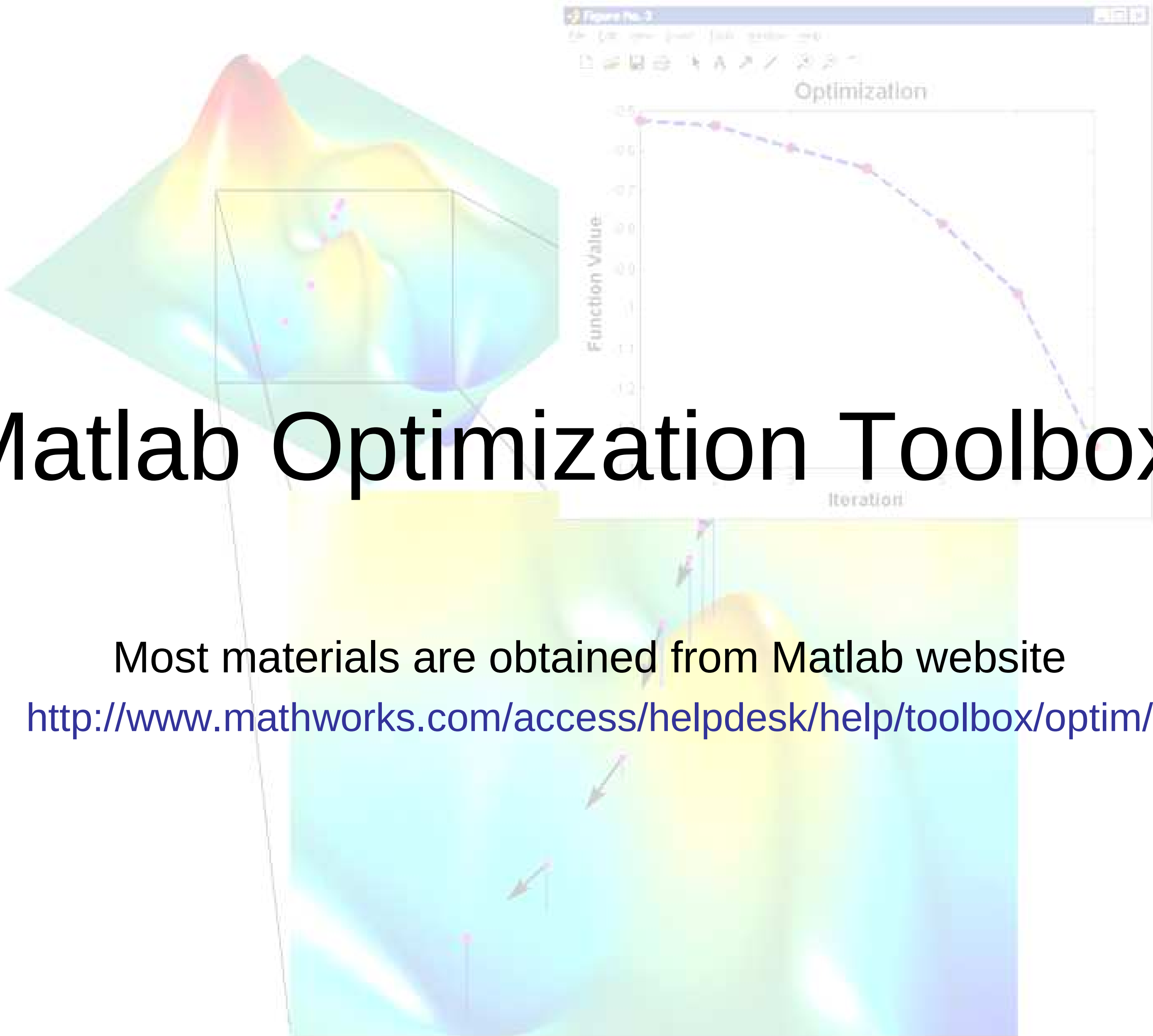


Matlab Optimization Toolbox

Most materials are obtained from Matlab website

http://www.mathworks.com/access/helpdesk/help/toolbox/optim/

What it can solve?

- Unconstrained nonlinear minimization
- Constrained nonlinear minimization
- Quadratic and linear programming
- Nonlinear least-squares and curve fitting
- Constrained linear least squares
- Sparse and structured large-scale problems, including linear programming and constrained nonlinear minimization
- Multiobjective optimization

Function List (I)

- **Unconstrained minimization**
  - fminunc Find minimum of unconstrained multivariable function
  - fminsearch Find minimum of unconstrained multivariable function using derivative-free method

- **Constrained minimization**
  - fminbnd Find minimum of single-variable function on fixed interval
  - Linprog Solve linear programming problems
  - quadprog Solve quadratic programming problems
  - fmincon Find minimum of constrained nonlinear multivariable fn
  - fminimax Solve minimax constraint problem
  - bintprog Solve binary integer programming problems
  - fgoalattain Solve multiobjective goal attainment problems
  - fseminf Find minimum of semi-infinitely constrained multivariable nonlinear function
  - ktrlink Find minimum of constrained or unconstrained nonlinear multivariable function using KNITRO third-party libraries

Function List (II)

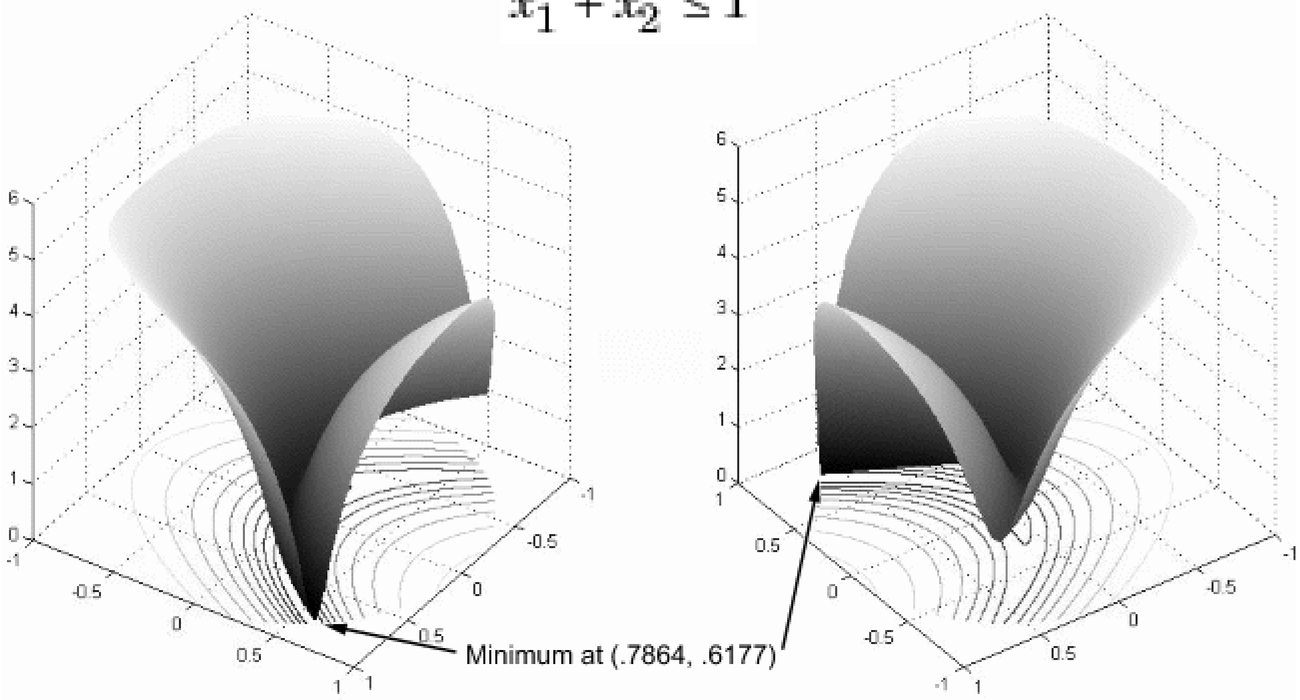
- **Equation Solving**
  - fsolve Solve system of nonlinear equations
  - fzero Find root of continuous function of one variable
- **Least Squares (Curve Fitting)**
  - lsqcurvefit Solve nonlinear curve-fitting (data-fitting) problems in least-squares sense
  - lsqlin Solve constrained linear least-squares problems
  - lsqnonlin Solve nonlinear least-squares (nonlinear data-fitting) problems
  - lsqnonneg Solve nonnegative least-squares constraint problem
- **GUI**
  - optimtool Tool to select solver, optimization options, and run problems
- **Utilities**
  - fzmult Multiplication with fundamental nullspace basis
  - gangstr Zero out "small" entries subject to structural rank
  - optimget Optimization options values
  - optimset Create or edit optimization options structure

How to use them?

- Example: Rosenbrock's function

$$f(x) = 100\left(x_2 - x_1^2\right)^2 + (1 - x_1)^2,$$

$$x_1^2 + x_2^2 \le 1$$



Use fmincon

The interface of fmincon

$$\min_x f(x) \text{ such that } \begin{cases} c(x) \le 0 \\ ceq(x) = 0 \\ A \cdot x \le b \\ Aeq \cdot x = beq \\ lb \le x \le ub, \end{cases}$$

```
x = fmincon(fun,x0,A,b,Aeq,beq,…
            lb,ub,nonlcon,options)
```

Write the objective function

```
function f = rosenbrock(x)
  f = 100*(x(2)-x(1)^2)^2+(1-x(1))^2;
```

Write the constraint

```
function [c, ceq] = unitdisk(x)
  c = x(1)^2 + x(2)^2 - 1;
  ceq = [ ];
```

Execution

```
[x,fval] = fmincon(@rosenbrock,[0 0],...
  [],[],[],[],[],[],@unitdisk)
```

Add Options

- Matlab does have ‘struct’
- Options is a huge structure containing
  - Algorithm: Chooses the algorithm used by the solver.
  - Display: Level of display.
  - GradObj: User-defined gradients for the objective functions.
  - Hessian: User-defined Hessian or Hessian information.
  - HessMult: Handle to a user-supplied Hessian multiply function.
  - HessUpdate: Quasi-Newton updating scheme.
  - Jacobian: User-defined Jacobian or Jacobian information.
  - JacobMult:User-defined Jacobian multiply function.
  - MaxIter: Maximum number of iterations allowed
  - TolFun: Termination tolerance on the function value.
  - …

Add Options

- Use command to set/get options

```
Options = optimset('Display','iter',...
        'Algorithm','active-set');
```

- Or just

```
Options = optimset;
Options.Display = 'iter';
Options.Algorithm = 'active-set';
```

  - Optimset can help validating the value.
- Or you can use GUI optimtool to set them.

Gradient

- If gradient or Hessian are not provided, Matlab uses finite difference to approximate them (for some functions).
- To provide gradient
  - Enable options: `optimset('GradObj','on')`
  - The user function

```
function [f g] = rosenbrock(x)
  f = 100*(x(2) - x(1)^2)^2 + (1-x(1))^2;
  g = [-400*(x(2)-x(1)^2)*x(1)-2*(1-x(1));
        200*(x(2)-x(1)^2)];
end
```

Algorithms and Hessian

- There are three algorithms in `fmincon`
  1. *Active-set*: use quasi-Newton approximation
  2. *Trust-region-reflective* (default): user supplied or finite-difference approximation
  3. *Interior-point*: many ways to define Hessian
     - User-supplied Hessian:

`optimset('Hessian','user-supplied','HessFcn',@hessianfcn)`

     - Quasi-Newton: `optimset('Hessian','bfgs')` or `optimset('Hessian',{'lbfgs',positive integer});`
     - Finite differences of the gradient

Option `HessMult`

- You can define your own matrix-vector multiplication function for Hessian

```
optimset('Hessian','user-supplied',...
         'SubproblemAlgorithm','cg', ...
         'HessMult',@HessMultFcn);
```

- In the trust-region-reflective algorithm

```
W = HessMultFcn(H,v);
```

- In the interior point algorithm

```
W = HessMultFcn(x,lambda,v);
```