Outline

- Optimization
  - `fminunc`, `fmincon`, `fminsearch`
  - Constrained optimization
  - Example: Log-likelihood function
MATLAB provides an efficient optimization toolbox that contains several routines to find the minimum of an user-supplied function.

The basic algorithm is based on the Newton-Raphson procedure, to solve constrained and unconstrained continuous and discrete problems.

The Newton-Raphson procedure is based on the iterative numerical evaluation of the gradient and the Hessian of the objective function with respect to the vector of variables.
Stopping criteria

The algorithm converges to a solution when one of the following conditions is satisfied:

- **TolFun**: is the termination tolerance placed on the objective function. Successful convergence occurs when the objective function value changes by less than TolFun. Default $1e^{-06}$.

- **TolX**: is the termination tolerance placed on the estimated parameter values. Similar to TolFun, successful convergence occurs when the parameter values change by less than TolX. Default $1e^{-06}$.

Note that the algorithms are designed to find the minimum of a function. If the objective function, $F(\theta)$, has to be maximized, then the function $C(\theta) = -F(\theta)$ will be passed to the optimizer, since the the minimum of $C(\theta)$ is located the same point as the maximum of $F(\theta)$. 
The optimization toolbox is designed in order to find the minimum of a user supplied function, that needs to be created in a .m file. For example, suppose we want to evaluate the minimum of the following function

$$f(x) = x^4 - 3x^3 + x^2 + 12;$$

(1)

First, we have to create a .m function, with the name of the function:

```matlab
function z = f_4(x);
z=x.^4+x.^2-3*x.^3+12;
```
Once, that the function is created and loaded into MATLAB, we can evaluate it, using one of the functions of the optimization toolbox. For example,

```matlab
x0=1;
options=optimset('Display','iter');
[x_star, f_star]=fminunc('f_4',x0,options);
```

The optimizer is able to minimize functions of more than one variable, as

```matlab
function z = fun_squares(x);
z=(x(1).^2+x(2).^2);
```
The command `fminunc` performs unconstrained optimization based on the numerical evaluation of the gradient. The syntax is:

```
[x,fval,exitflag]=fminunc('fun',x0,options,var_1,var_2,...)
```

where `x0` is the vector of starting values, `fun` is the function to be minimized, `options` is a vector of strings that contains the optimization options, while `var_1, var_2, ...` are the remaining inputs of the function `fun`. Typically, in economic applications these variables are the data.
The output terms are: $x$, that are the variables at the optimum, $fval$, that is the value of the function evaluated at the optimum, $exitflag$, that is a flag that determines if the convergence to the optimal values was successful or not.
The command `fminsearch` also minimizes a function of several variables. The numerical method is derivative-free, i.e. it is not based on gradient evaluation. This algorithm uses the simplex search Nelder-Mead method that is a nonlinear optimization technique based on the concept of simplex. The syntax is

```matlab
[x,fval,exitflag]=fminsearch('fun',x0,options,var_1,var_2,...)
```
Example with the Rosenbrock function:

```matlab
function z=banana(x);
z=100*(x(2)-x(1)^2)^2+(1-x(1))^2;
```

that has a minimum in [1, 1].
The problem

In many applications, we face optimization problems under certain constraints. For example, we want to minimize the variance of a portfolio of risky assets,

\[
\begin{align*}
    w^* &= \arg \min_w (w' \Sigma w) \\
    w_i > 0 \\
    \iota' w &= 1
\end{align*}
\]  

(2)

Problem like this can be solved numerically modifying the standard optimization routines by inserting disincentives of exiting from the admissible region.
The problem

In general the constrained optimization problem can be defined as

\[
\begin{align*}
\min_x f(x) \\
\text{s.t.} & \quad g(x) \leq 0
\end{align*}
\]

The most common techniques to transform a constrained problem into an unconstrained problem are:

- **Penalization functions**
- **Barrier functions**
Penalization functions

The objective function is transformed as

$$\min_x (f(x) + \gamma \cdot \|\max[g(x), 0]\|^2) \quad \gamma > 0$$  \hspace{1cm} (4)

such that every time the function exits from the boundary, its value is increased by an amount that is proportional to the distance from the admissible set. Hence, the optimization problem becomes unconstrained on the function

$$\min_x f_p(x, \gamma_h)$$  \hspace{1cm} (5)

for an increasing sequence of positive values of $\gamma_h$. The algorithm stops when

$$\|\hat{x}^h - \hat{x}^{h-1}\| < \epsilon_\gamma \quad \epsilon_\gamma > 0.$$  \hspace{1cm} (6)
The objective function in this case is transformed as

$$\min_x \left( f(x) - \gamma \sum_{i=1}^{m} \frac{1}{g_i(x)} \right) \quad \gamma > 0 \quad (7)$$

we impose a very large cost on feasible points that lie close to the boundary since $-\frac{1}{g_i(x)} \to \infty$ as $g_i(x) \to 0^-$. Hence, the optimization problem becomes unconstrained on the function

$$\min_x f_b(x, \gamma) \quad (8)$$

for an increasing sequence of positive values of $\gamma$. The algorithm stops when

$$||\hat{x}^h - \hat{x}^{h-1}|| < \epsilon_{\gamma} \quad \epsilon_{\gamma} > 0. \quad (9)$$
This command performs constrained optimization, in order to find the minimum of a constrained nonlinear multi-variable function. The syntax is:

\[
[x,fval,exitflag]=\text{fmincon}(’\text{fun’},x0,A,b,A_{eq},b_{eq},lb,ub,\text{nonlcon},\text{options},\text{var}_1,\text{var}_2,\ldots)
\]

where \(x0\) is the vector of starting values, ‘\text{fun}’ is the function to be minimized, \text{options} is a vector of strings that contains the optimization options, while \text{var}_1,\text{var}_2,\ldots\) are the remaining inputs of the function ‘\text{fun}’.
A and b denote the equality constraints, such that $A_{eq}x = b_{eq}$, while $A$ and $b$ stand for the inequality constraints, such that $Ax \leq b$. $lb$ and $ub$ are the lower and upper bounds constraints on the parameters. The input `nonlcon` is an handle to an user supplied function with non linear constraints.
The output terms are: $x$, that are the variables at the optimum, $fval$, that is the value of the function evaluated at the optimum, $exitflag$, that is a flag that determines if the convergence to the optimal values was successful or not.
The optimization problem is set as:

\[ \text{lb} = [0 \ 0]; \]
\[ \text{ub} = [2 \ 2]; \]
\[ \text{options} = \text{optimset}('\text{Display}', '\text{iter}') ; \]
\[ [\text{x}\_\text{star1}, \text{f}\_\text{star1}, \text{eflag1}] = \text{fmincon}('\text{banana}', \text{x}, [], [], [], [], \text{lb}, \text{ub}, [], \text{options}); \]

where \( x_1 \) and \( x_2 \) are restricted to be in the interval \([0,2]\).
Inequality constraints

Alternatively we can impose equality or inequality constraints. For example

```matlab
function z=CES(X,F,r,lambda);
z=-F*(lambda*X(1)^r+(1-lambda)*X(2)^r)^(1/r);
```

is called and minimized by `fmincon`. 

Constrained optimization

Inequality-equality constraints

\[ F = 100; \]
\[ r = 0.1; \]
\[ X = [0.1 \ 0.9]; \]
\[ \text{options} = \text{optimset}('Display', 'iter'); \]
\[ A = [-1 \ 0; \ 0 \ -1]; \]
\[ b = [0 \ 0]; \]
\[ \text{Aeq} = [1 \ 1]; \]
\[ \text{beq} = [1]; \]
\[ [x\_star, f\_star, exitflag] = \text{fmincon}('CES', X, A, b, Aeq, beq, [], [], [], options, F, r, lambda); \]
Exercise

- Generate iid observations from a Gaussian distribution with mean $\mu$ and variance $\sigma^2$;
- Write a function that computes the log-likelihood function of iid random variables with Gaussian distribution with mean $\mu$ and variance $\sigma^2$;
- Estimate $\hat{\mu}$ and $\hat{\sigma}^2$ by maximum likelihood using the data obtained from the simulated trajectory at point 1.
- Try to use both unconstrained and constrained optimization routines.