Solving economic problems with MATLAB

Antonio Mele

University of Surrey

Why numerical methods?

,

- Many economic problems are VERY complicated

Why numerical methods?

,

- Many economic problems are VERY complicated
- No analytical solution

Why numerical methods?

,

- Many economic problems are VERY complicated
- No analytical solution
- Often, even if we can describe some QUALITATIVE features, we need numerical methods for QUANTITATIVE results

Why numerical methods?

,

- Many economic problems are VERY complicated
- No analytical solution
- Often, even if we can describe some QUALITATIVE features, we need numerical methods for QUANTITATIVE results
- Mostly: solving non-linear equations and optimisation problems

Optimization Toolbox: basics

The toolbox includes:

- linear programming

Optimization Toolbox: basics

The toolbox includes:

- linear programming
- quadratic programming

Optimization Toolbox: basics

The toolbox includes:

- linear programming
- quadratic programming
- binary integer programming

Optimization Toolbox: basics

The toolbox includes:

- linear programming
- quadratic programming
- binary integer programming
- nonlinear optimization

Optimization Toolbox: basics

The toolbox includes:

- linear programming
- quadratic programming
- binary integer programming
- nonlinear optimization
- nonlinear least squares

Optimization Toolbox: basics

The toolbox includes:

- linear programming
- quadratic programming
- binary integer programming
- nonlinear optimization
- nonlinear least squares
- systems of nonlinear equations

Optimization Toolbox: basics

The toolbox includes:

- linear programming
- quadratic programming
- binary integer programming
- nonlinear optimization
- nonlinear least squares
- systems of nonlinear equations
- multiobjective optimization

Optimization Toolbox: basics (cont.)

Solvers are minimizers!

Optimization Toolbox: basics (cont.)

Solvers are minimizers!
If you perform a **maximization**: define your objective as $-f(x)$ and remember to change sign to the value obtained.

Optimization Toolbox: basics (cont.)

Solvers are minimizers!
If you perform a **maximization**: define your objective as $-f(x)$ and remember to change sign to the value obtained.
All the functions are **function functions**: they take other functions as inputs. In particular:

- the objective function is a **function** file
- nonlinear constraints must be set in a **function** file

Optimization Toolbox: basics (cont.)

Solvers are minimizers!
If you perform a **maximization**: define your objective as $-f(x)$ and remember to change sign to the value obtained.
All the functions are **function functions**: they take other functions as inputs. In particular:

- the objective function is a **function** file
- nonlinear constraints must be set in a **function** file

Options: set with `optimset`

REMINDER: what is a function?

```
function z = olscoefficient(X,Y)
  z = inv(X'*X)*(X'*Y);
end
```

Nonlinear equations in one variable

Use the function `fzero`: finds roots of continuous functions

Syntax:

```
[x, fval] = fzero('objfun',x0);
```

`x`: optimum

`fval`: value of the objective function calculated in the optimum

`objfun`: function file where we have stored the objective function

`x0`: initial condition from which `fzero` looks for a solution

Nonlinear equations in more than one variable

If you have n nonlinear equations $F_i(x) = 0$, with $x \in R^n$, use `fsolve`.
Syntax:

```
    [x, fval] = fsolve('objfun',x0);
```

`x`: optimum
`fval`: value of the objective function calculated in the optimum
`objfun`: function file where we have stored the objective function
`x0`: initial condition from which `fsolve` looks for a solution

Example 1

Solve the following system of equations:

$$c_1^{-\sigma} = \beta(1+r)c_2^{-\sigma}$$
$$c_1 + \frac{c_2}{1+r} = y_1 + \frac{y_2}{1+r}$$

where $r = 0.05$, $\sigma = 2$, $\beta = .99$, and $y_1 = y_2 = 1$.

Example 2

Same as before, but with an initial amount of savings s_0:

$$c_1^{-\sigma} = \beta(1+r)c_2^{-\sigma}$$
$$c_1 + \frac{c_2}{1+r} = y_1 + \frac{y_2}{1+r} + s_0$$

where $r = 0.05$, $\sigma = 2$, $\beta = .99$, and $y_1 = y_2 = 1$.
Solve for the optimal allocation for different values of the initial savings.

Optimization: general rules

Types of constraints:

Optimization: general rules

Types of constraints:

1. **Bound Constraints**: $x \geq l$, $x \leq u$

Optimization: general rules

Types of constraints:

1. **Bound Constraints**: $x \geq l$, $x \leq u$
2. **Linear Inequality Constraints**: $A \cdot x \leq b$, where A is an m by n matrix, which represents m constraints for an n dimensional vector x, and b is m dimensional

Optimization: general rules

Types of constraints:

1. **Bound Constraints**: $x \geq l$, $x \leq u$
2. **Linear Inequality Constraints**: $A \cdot x \leq b$, where A is an m by n matrix, which represents m constraints for an n dimensional vector x, and b is m dimensional
3. **Linear Equality Constraints**: $Aeq \cdot x = beq$, same dimensionality of linear inequality constraints

Optimization: general rules

Types of constraints:

1. **Bound Constraints**: $x \geq l$, $x \leq u$
2. **Linear Inequality Constraints**: $A \cdot x \leq b$, where A is an m by n matrix, which represents m constraints for an n dimensional vector x, and b is m dimensional
3. **Linear Equality Constraints**: $Aeq \cdot x = beq$, same dimensionality of linear inequality constraints
4. **Nonlinear Constraints**: $c(x) \leq 0$ and $ceq(x) = 0$. Both c and ceq are scalars or vectors representing several constraints

Setting options

Each algorithm has many options on the type of algorithm to use, on the output to show in command window, the convergence criterion, etc.

Setting options

Each algorithm has many options on the type of algorithm to use, on the output to show in command window, the convergence criterion, etc.
To set them: use `optimset`

```
options = optimset('param1',value1, 'param2',value2,...);
```

Setting options

Each algorithm has many options on the type of algorithm to use, on the output to show in command window, the convergence criterion, etc.
To set them: use `optimset`

```
options = optimset('param1',value1, 'param2',value2,...);
```

IMPORTANT: some parameter values are strings, therefore you have to enter them between ' '. Example:

```
options  = optimset('Display','iter');
```

Unconstrained Minimization

No constraints

$$\min_x f(x)$$

Unconstrained Minimization

No constraints

$$\min_x f(x)$$

Use `fminunc`:

```
[x, fval] = fminunc('objfun',x0)
```

Constrained Minimization

$$\begin{aligned} \min_{x} \quad & f(x) \\ s.t. \quad & x \geq LB, \quad x \leq UB \\ & A \cdot x \leq B, \quad Aeq \cdot x = Beq \\ & c(x) \leq 0, \quad ceq(x) = 0 \end{aligned}$$

Constrained Minimization

$$
\begin{aligned}
\min_{x} \quad & f(x) \\
s.t. \quad & x \geq LB, \quad x \leq UB \\
& A \cdot x \leq B, \quad Aeq \cdot x = Beq \\
& c(x) \leq 0, \quad ceq(x) = 0
\end{aligned}
$$

Use `fmincon`:

```
[x, fval] = fmincon('objfun',x0,A,B,Aeq,Beq,...
    LB,UB,nonlcon);
```

When one or more constraints absent: use []

Writing a nonlinear constraint function

Writing a nonlinear constraint function

It must have a particular structure

```
function [c, ceq] = nonlinconst(input1,input2,...)

c(1) = ...
c(2) = ...
...
ceq(1) = ...
ceq(2) = ...
...
```

Writing a nonlinear constraint function

It must have a particular structure

```
function [c, ceq] = nonlinconst(input1,input2,...)

c(1) = ...
c(2) = ...
...
ceq(1) = ...
ceq(2) = ...
...
```

If no constraints of one type: use `ceq = [];`

Example 1

Use fmincon to maximize the utility function $u(c) = \frac{c^{1-\sigma}}{1-\sigma}$ under the constraints:

$$c \geq 0$$
$$c \leq y$$

where $\sigma = 2$ and $y = 1$.

Example 2

Take again a two-periods economy with an initial amount of savings s_0. Your problem is

$$\max_{c_1,c_2} \frac{c_1^{1-\sigma}}{1-\sigma} + \beta \frac{c_2^{1-\sigma}}{1-\sigma}$$
$$s.t. \quad c_1 + \frac{c_2}{1+r} \leq y_1 + \frac{y_2}{1+r} + s_0$$

where $r = 0.05$, $\sigma = 2$, $\beta = .99$, and $y_1 = y_2 = 1$. Solve for the optimal allocation for different values of the initial savings. (Hint: vectorizing the procedure is not possible here, therefore you need to use a `for` loop).

Example 3

Maximize the utility function $u(c_1, ..., c_{10}) = \sum_{i=1}^{10} \frac{c_i^{1-\sigma}}{1-\sigma}$ under the constraints:

$$c_i \geq 0 \text{ for all } i$$
$$\sum_{i=1}^{10} c_i \leq y$$
$$2c_3 + c_1 = 12$$
$$0.5(c_5 - c_4)^2 = 4$$

where $\sigma = 2$ and $y = 100$.