Simple Numerical Optimization Algorithms

Felix Mauersberger

Rheinische Friedrich-Wilhelms-Universität Bonn

July 3, 2019

Introduction

Optimization

- typical problem in economics: find an optimum
- Case 1:

$$\max_x f(x) = 0.5x - 0.02x^2$$

easy: just solve for x satisfying $f'(x) = 0$

- Case 2:

$$\max_x f(x) = 0.5\ln(x) - 0.02x^2 + \frac{1}{(1+x)^{\frac{5}{3}}}$$

can't solve analytically $\Rightarrow$ need to use numerical methods

Numerical optimization

- continuous differentiable objective function: derivative-based methods
- arbitrary objective function: derivative-free methods

Numerical optimization in MATLAB

Numerical optimization in MATLAB

Three functions in MATLAB's Optimization Toolbox:

- **fminunc:** unconstrained optimization, derivative-based
- **fmincon:** constrained optimization, derivative-based
- **fminsearch:** unconstrained optimization, derivative-free

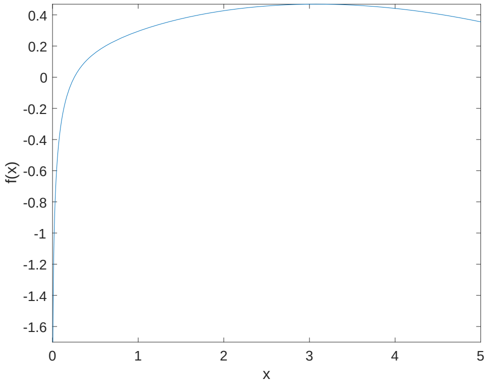
fminunc

- finds parameters that minimize given function
- Syntax:
  `x=fminunc(objfunc,x0)`
    - `x0`: starting guess
    - `objfun`: objective function specified in a different script
    - `x`: solution

fminunc

Example 1:

$$\max_x f(x) = 0.5 \ln x - 0.02x^2 + \frac{1}{(1+x)^{\frac{5}{3}}}$$



fminunc

Example 1:

$$\max_x f(x) = 0.5 \ln x - 0.02x^2 + \frac{1}{(1+x)^{\frac{5}{3}}}$$

Step 1: Main code

```
^^Ix0=1;
^^Ix=fminunc(@objfun,x0)
^^I
```

Step 2: function

```
^^Ifunction y=objfun(x)
^^Iy=0.5*log(x)-0.02 x^2+(1+x)^(-5/3) ;
^^Iy=-y;
^^Iend
^^I
```

fminunc

fminunc uses the BFGS Quasi-Newton method:

$$x_{k+1} = x_k + \alpha_k p_k$$

1. Compute direction p_k

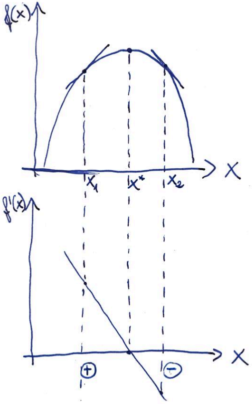
$$\begin{aligned} p_k =& - B_k^{-1} \nabla f(x_k) \\ B_{k+1} =& B_k - \frac{B_k s_k s_k' B_k}{s_k' B_k s_k} + \frac{y_k y_k'}{y_k' s_k} \end{aligned}$$

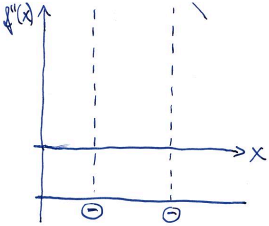
where $s_k \equiv x_{k+1} - x_k$ and $y_k \equiv f_{k+1} - f_k$

2. Compute stepsize α_k. Use linesearch to approximately solve

$$\min_{\alpha_k} f(x_k + \alpha_k p_k)$$

BFGS Quasi-Newton method





fminunc

Introduce options:

```
^^Ioptions=optimset('Opt1',Opt1Val,'Opt2',Opt2Val,...)
^^I
```

Important algorithm options:

- Display:
  `optimset('Display','off')`
- Tolerance level:
  `optimset('TolFun',1e-6)`
  `optimset('TolX',1e-6)`
- Maximum number of iterations or evaluations:
  `optimset('MaxIter',400)`
  `optimset('MaxFunEvals',100)`

fminunc

Caution:

- Objective function must be continuous
- Local minima and flat regions are big issues

fmincon

- Finds parameters that minimize a given function, subject to a constraint
- Same algorithm as fminunc
- Syntax:

```
^^I^^Ix=fmincon(objfun,x0,A,b,Aeq,beq,lb,ub,nlcon,optic
^^I^^I
```

  - `x0`: starting guess
  - `objfun`: objective function specified in a different script
  - `A,b`: linear inequality constraint: $Ax \leq b$
  - `Aeq, beq`: linear equality constraint: $Ax = b$
  - `lb,ub`: lower and upper bounds: $x \in [lb, ub]$
  - `nlcon`: nonlinear constraint
  - `x`: solution

fminsearch

- Finds parameters that minimize a given function
- Derivative-free algorithm: ...

fminsearch

- Finds parameters that minimize a given function
- Derivative-free algorithm: ...

Nelder-Mead

fminsearch

- Finds parameters that minimize a given function
- Derivative-free algorithm: ...

Nelder-Mead

- Syntax:

```
^^I^^Ix=fminsearch(objfun,x0,options)
^^I^^I
```

  - x0: starting guess
  - objfun: objective function specified in a different script
  - x: solution

The Nelder-Mead algorithm

1. select two points: x_1 and x_2

The Nelder-Mead algorithm

1. select two points: x_1 and x_2
2. order so that $\hat{x} = x_1$ and $\bar{x} = x_2$ if $f(\hat{x}) \geq f(\bar{x})$; and $\hat{x} = x_2$ and $\bar{x} = x_1$ otherwise

The Nelder-Mead algorithm

1. select two points: x_1 and x_2
2. order so that $\hat{x} = x_1$ and $\bar{x} = x_2$ if $f(\hat{x}) \geq f(\bar{x})$; and $\hat{x} = x_2$ and $\bar{x} = x_1$ otherwise
3. **reflection:** $x^R = \hat{x} + (\hat{x} - \bar{x})$

The Nelder-Mead algorithm

1. select two points: x_1 and x_2
2. order so that $\hat{x} = x_1$ and $\bar{x} = x_2$ if $f(\hat{x}) \geq f(\bar{x})$; and $\hat{x} = x_2$ and $\bar{x} = x_1$ otherwise
3. **reflection:** $x^R = \hat{x} + (\hat{x} - \bar{x})$
4. if $f(x^R) > f(\hat{x})$ and $f(x^R) > f(\bar{x})$, **expansion**:
   $x^E = \hat{x} + \alpha(x^R - \hat{x})$, $\alpha > 1$
   if $f(x^R) \leq f(\hat{x})$ and $f(x^R) > f(\bar{x})$, update $\bar{x} = x^R$ and go to step 2
   if $f(x^R) \leq f(\hat{x})$ and $f(x^R) \leq f(\bar{x})$, **contraction**:
   $x^C = \hat{x} + \beta(\bar{x} - \hat{x})$, $\beta < \frac{1}{2}$

The Nelder-Mead algorithm

1. select two points: x_1 and x_2
2. order so that $\hat{x} = x_1$ and $\bar{x} = x_2$ if $f(\hat{x}) \geq f(\bar{x})$; and $\hat{x} = x_2$ and $\bar{x} = x_1$ otherwise
3. **reflection:** $x^R = \hat{x} + (\hat{x} - \bar{x})$
4. if $f(x^R) > f(\hat{x})$ and $f(x^R) > f(\bar{x})$, **expansion**: $x^E = \hat{x} + \alpha(x^R - \hat{x})$, $\alpha > 1$
   if $f(x^R) \leq f(\hat{x})$ and $f(x^R) > f(\bar{x})$, update $\bar{x} = x^R$ and go to step 2
   if $f(x^R) \leq f(\hat{x})$ and $f(x^R) \leq f(\bar{x})$, **contraction**: $x^C = \hat{x} + \beta(\bar{x} - \hat{x})$, $\beta < \frac{1}{2}$
5. if $f(x^E) > f(x^R)$, update $\bar{x} = x^E$ and go to step 2
   if $f(x^E) < f(x^R)$, update $\bar{x} = x^R$ and go to step 2

The Nelder-Mead algorithm

1. select two points: x_1 and x_2
2. order so that $\hat{x} = x_1$ and $\bar{x} = x_2$ if $f(\hat{x}) \geq f(\bar{x})$; and $\hat{x} = x_2$ and $\bar{x} = x_1$ otherwise
3. **reflection:** $x^R = \hat{x} + (\hat{x} - \bar{x})$
4. if $f(x^R) > f(\hat{x})$ and $f(x^R) > f(\bar{x})$, **expansion**:
   $x^E = \hat{x} + \alpha(x^R - \hat{x})$, $\alpha > 1$
   if $f(x^R) \leq f(\hat{x})$ and $f(x^R) > f(\bar{x})$, update $\bar{x} = x^R$ and go to step 2
   if $f(x^R) \leq f(\hat{x})$ and $f(x^R) \leq f(\bar{x})$, **contraction**:
   $x^C = \hat{x} + \beta(\bar{x} - \hat{x})$, $\beta < \frac{1}{2}$
5. if $f(x^E) > f(x^R)$, update $\bar{x} = x^E$ and go to step 2
   if $f(x^E) < f(x^R)$, update $\bar{x} = x^R$ and go to step 2
6. if $f(x^C) > f(\bar{x})$, update $\bar{x} = x^C$ and go to step 2
   if $f(x^C) < f(\bar{x})$, **shrink**: replace $\bar{x} = \hat{x} + \gamma(\bar{x} - \hat{x})$ and go to step 2, $\gamma < 1$

The Nelder-Mead algorithm

1. select two points: x_1 and x_2
2. order so that $\hat{x} = x_1$ and $\bar{x} = x_2$ if $f(\hat{x}) \geq f(\bar{x})$; and $\hat{x} = x_2$ and $\bar{x} = x_1$ otherwise
3. **reflection:** $x^R = \hat{x} + (\hat{x} - \bar{x})$
4. if $f(x^R) > f(\hat{x})$ and $f(x^R) > f(\bar{x})$, **expansion**: $x^E = \hat{x} + \alpha(x^R - \hat{x})$, $\alpha > 1$
   if $f(x^R) \leq f(\hat{x})$ and $f(x^R) > f(\bar{x})$, update $\bar{x} = x^R$ and go to step 2
   if $f(x^R) \leq f(\hat{x})$ and $f(x^R) \leq f(\bar{x})$, **contraction**: $x^C = \hat{x} + \beta(\bar{x} - \hat{x})$, $\beta < \frac{1}{2}$
5. if $f(x^E) > f(x^R)$, update $\bar{x} = x^E$ and go to step 2
   if $f(x^E) < f(x^R)$, update $\bar{x} = x^R$ and go to step 2
6. if $f(x^C) > f(\bar{x})$, update $\bar{x} = x^C$ and go to step 2
   if $f(x^C) < f(\bar{x})$, **shrink**: replace $\bar{x} = \hat{x} + \gamma(\bar{x} - \hat{x})$ and go to step 2, $\gamma < 1$

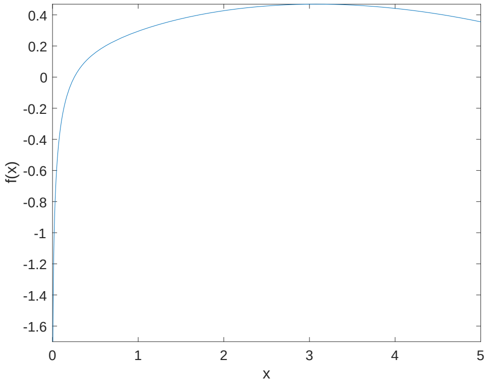
Iterate until convergence: $\hat{x} \approx \bar{x}$ and/or $f(\hat{x}) \approx f(\bar{x})$

Examples

fminsearch

Example 1:

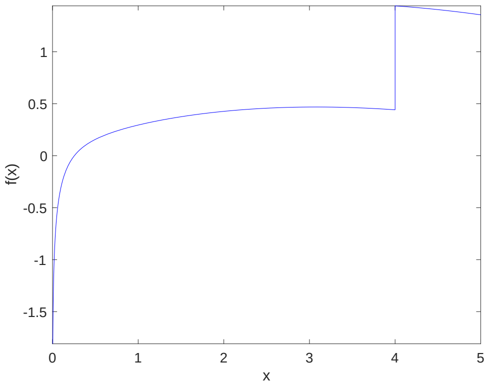
$$\max_x f(x) = 0.5 \ln x - 0.02x^2 + \frac{1}{(1+x)^{\frac{5}{3}}}$$



fminsearch

Example 2: Discontinuous objective function:

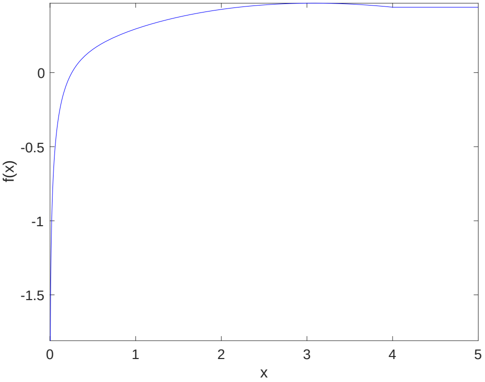
$$\max_x f(x) = \begin{cases} 0.5 \ln x - 0.02x^2 + \frac{1}{(1+x)^{\frac{5}{3}}} & 0 < x \leq 4 \\ 0.5 \ln x - 0.02x^2 + \frac{1}{(1+x)^{\frac{5}{3}}} & 4 < x \end{cases}$$



fminsearch

Example 3: Objective function with flat regions:

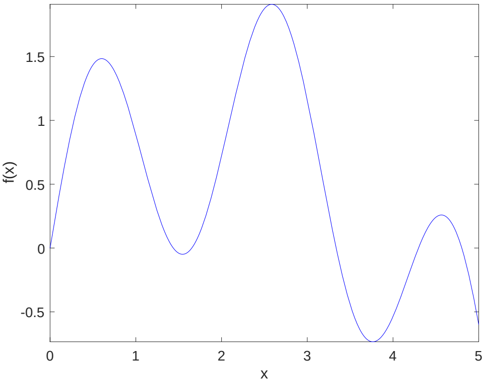
$$\max_x f(x) = \begin{cases} 0.5\ln x - 0.02x^2 + \frac{1}{(1+x)^{\frac{5}{3}}} & 0 < x \leq 4 \\ 0.441546 & 4 < x \end{cases}$$



fminsearch

Example 4: Objective function with local maxima:

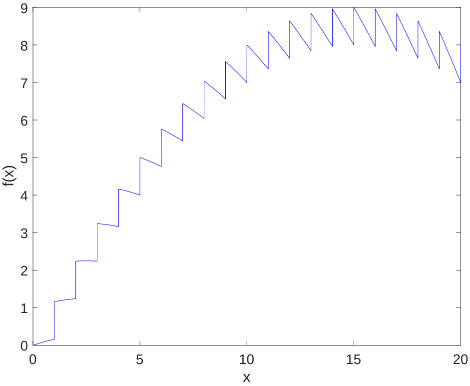
$$\max_x f(x) = \sin(3x) + x - 0.25x^2$$



fminsearch

Example 5: Objective function with many discontinuities:

$$\max_{x} f(x) = \lfloor x \rfloor + 0.2x - 0.04x^2$$



Conclusion

fminsearch vs. fminunc

- Rule of thumb: use fminunc for continuous differentiable functions, fminsearch otherwise
- No option guarantees that you find a (global) optimum
- For complicated cases
  - understand the problem well
  - try different algorithms
  - try different starting points