Numerical Method—Optimization

(Jing Li, Miami University)

Why Numerical Method?

1. There are problems that we know the solution exists, but we cannot find its analytical or closed form
2. For example, the root for the equation

$$\frac{1}{x} = \sin x$$

   exists because the two curves cross at some points. But we cannot show $x =?$

3. This implies that we cannot analytically optimize

$$\log x + \cos x$$

   because the first order condition (taking derivative) is just $\frac{1}{x} = \sin x$. Nevertheless, we know the optimum exists somewhere.

4. Numerical method can be used to solve this kind of problem, and many others (such as when we do not want to take derivatives).

R Code of Numerically Solving $\frac{1}{x} = \sin x$

```
# define the gap or distance function
d = function(x)
{return(1/x-sin(x))}
# find the root between 2 and 4 (why those two values?---Draw graph)
i = 0; smallx = 2; bigx = 4
while (i<20) {
root = (smallx + bigx)/2
if (d(root)*d(bigx)>0)
{bigx = root}
else
{smallx = root}
cat("--------------------------", "\n")
cat("Root is ", root, "\n")
i = i+1
}
```

Bisection Method to Solve Equation

1. In general we want to solve $f(x) = g(x)$. Let's define the gap (distance) function

   $$d(x) = f(x) - g(x)$$

   We want to find which x solves $d(x) = 0$

2. The algorithm of bisection method is, at beginning, pick any two values x^{small} and x^{big} that have <u>opposite</u> signs of d, i.e.,

   $$d(x^{small})d(x^{big}) < 0$$

   (a) next inside the loop, compute middle point (tentative root) in each iteration

   $$m = \frac{x^{small} + x^{big}}{2}$$

   (b) replace x^{big} with m if they have the <u>same</u> signs of d; otherwise replace x^{small} with m. Redo (a) and (b) repeatedly. Loop does not stop until the change in m is close to zero.

A Finance Example

1. Consider a three-year coupon bond with price $P = 95$, face value 100, and annual coupon rate 10%. We want to find internal rate of return (IRR), which can be used to compare different bonds

2. By definition, IRR solves this equation

$$P = \frac{10}{1+IRR} + \frac{10}{(1+IRR)^2} + \frac{10+100}{(1+IRR)^3}$$

3. Let $x = 1/(1+IRR)$, the equation can be rewritten as

$$110x^3 + 10x^2 + 10x - 95 = 0 \tag{1}$$

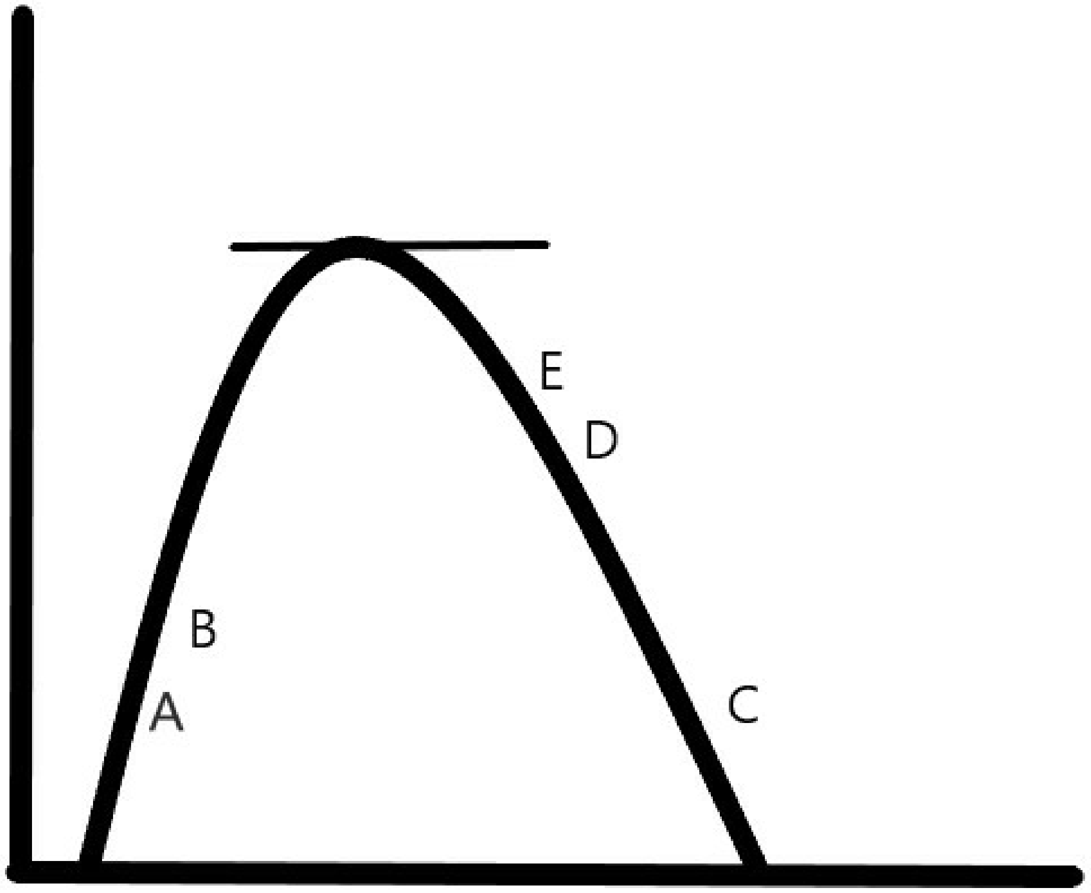
   There is no analytical solution for this third-order polynomial equation

4. Homework asks you to use bisection method to find x and IRR.

Main Ideas of Numerical Optimization

1. Suppose we want to maximize something such as utility or profit
2. This is like climbing a mountain
3. We get started somewhere (initial value), and we keep climbing (doing iteration) until reaching the top (a function is maximized)
4. Mathematically, the slope of the mountain is first derivative. It is uphill if the derivative is positive, and downhill if the derivative is negative. At the top, the derivative is zero (first order condition)
5. In short, the derivative tells us where to go (up or down) in next round of loop

How to Reach Top (Recursively)?



Iteration (algorithm)

1. Suppose we start with point A, where the slope (gradient, or first order derivative) is positive. Next we increase the value, move to point B, and continue...
2. On the other hand, if we start with point C, where the slope is negative, we should decrease the value, and move to point D. Then we treat point D as a new initial value, and move to point E in next round. The iteration does not stop until we reach the top (maximum) where the slope is zero
3. To summarize, to implement the numerical method, we need to know
   (a) The sign of change (increase or decrease). This is determined by the sign of the first derivative or gradient
   (b) The size of change, called step size. It will take long time for the algorithm to converge if the step size is too small, but overshooting can happen (moving from point D to point B, other than E) if the step size is too big
   (c) Stopping criterion. For instance, the algorithm may stop if the first derivative is sufficiently close to zero

Example

1. We want to maximize the function

$$f(x) = -x^2$$

2. The first order derivative, or gradient, is

$$f'(x) = -2x$$

3. The R codes of defining the function and its derivative are

```
f = function(x) {
return(-x^2)
}
fprime = function(x) {
return(-2*x)
}
```

4. By letting $f'(x) = 0$ (first order condition) we already know the function is maximized at $x^* = 0$. Next we show how an iterative algorithm <u>automatically</u> finds $x^* = 0$.

Example—Continued

First we set the initial value $x_0 = 5$, and the stopping criterion. Then we use a while loop to conduct the iteration

```
x0 = 5
tol = 0.001
while (abs(fprime(x0))>tol) {
step = 1
x1 = x0 + step*2*((fprime(x0)>0)-0.5)
while (f(x1)<f(x0)) {
step = step/10
x1 = x0 + step*2*((fprime(x0)>0)-0.5)
}
cat("New initial value is ", x1, "\n")
cat("Gradient is ", fprime(x1), "\n")
cat("f(x) is ", f(x1), "\n")
x0 = x1
}
```

Example—Remarks

1. The while loop does not stop until the first order derivative is very close to zero.
2. We first try the step size of 1
3. If the slope is positive, $x_1 = x_0 + 1$; if the slope is negative, $x_1 = x_0 - 1$
4. We divide the step size by 10 if overshooting occurs, i.e., when $f(x_1) < f(x_0)$
5. To monitor the progress, in each iteration, we print out the new initial value, the gradient, and the function value
6. In the end of loop, we let the new value be the initial value for the next iteration

Example—Result

```
New initial value is  4
Gradient is  -8
f(x) is  -16
New initial value is  3
Gradient is  -6
f(x) is  -9
New initial value is  2
Gradient is  -4
f(x) is  -4
New initial value is  1
Gradient is  -2
f(x) is  -1
New initial value is  0
Gradient is  0
f(x) is  0
```

Example—Remarks

1. There are two signals that the algorithm goes well
   (a) The gradient approaches zero: $f'(x) = -8, -6, \dots, 0$
   (b) The function value increases: $f(x) = -16, 19, \dots, 0$
2. The loop stops $\underline{\text{automatically}}$ when $x = 0$. This is the optimal value we try to find.
3. Exercise
   (a) Try the new initial value $x_0 = -10$ and see what happens
   (b) Try the step size of 1.5, and see what happens
   (c) Try the step size of 2, and see what happens
   (d) Change the stopping criterion to $tol = 2$ and see what happens
4. Lesson: $\underline{\text{Fine-tuning}}$ the algorithm is an art. The key is to find proper initial value, step size and stopping criterion. It takes patience, experience, and trial and errors.

Exercise

You may find this line hard to understand

```
x1 = x0 + step*2*((fprime(x0)>0)-0.5)
```

Instead, please write down the R `if...else` statement to do the following: if the slope is positive, $x_1 = x_0 + \texttt{step}$; if the slope is negative, $x_1 = x_0 - \texttt{step}$. (Hint: google “R, if”)

Newton's Method

1. It's easier to specify the step size if the second order derivative (called Hessian) $f''(x)$ is known. In practice, deriving the Hessian can be difficult.
2. Consider a Taylor expansion of $f'(x_1)$

$$f'(x_1) \approx f'(x_0) + f''(x_0)(x_1 - x_0) \tag{2}$$

   If x_1 optimizes the function, then $f'(x_1) = 0$, and it follows that

$$x_1 \approx x_0 - \frac{f'(x_0)}{f''(x_0)} \tag{3}$$

   and according to (3), the step size is $-\frac{f'(x_0)}{f''(x_0)}$.
3. More generally, to avoid overshooting, we can let

$$x_1 = x_0 - k\left(\frac{f'(x_0)}{f''(x_0)}\right) \tag{4}$$

   where the tuning parameter k can be 1, 0.1, 0.01...Typically the closer we are to x^*, the smaller is k.

Example—Continued

The R codes to apply the Newton's method to maximize $-x^2$ is

```
fprime2 = function(x) {
return(-2)
}
tol = 0.001; x0 = 5
while (abs(fprime(x0))>tol) {
step = -fprime(x0)/fprime2(x0)
x1 = x0 + step
while (f(x1)<f(x0)) {
step = step/10
x1 = x0 + step
}
cat("New value is ", x1, "\n")
cat("Gradient is ", fprime(x1), "\n")
x0 = x1
}
```

Derivative-Free Methods

So far the algorithm assumes the first order derivative $f'(x)$ is known and supplied. Alternatively, we can use (several) derivative-free methods. For instance, recall how the derivative is defined

$$f'(x) = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x} \tag{5}$$

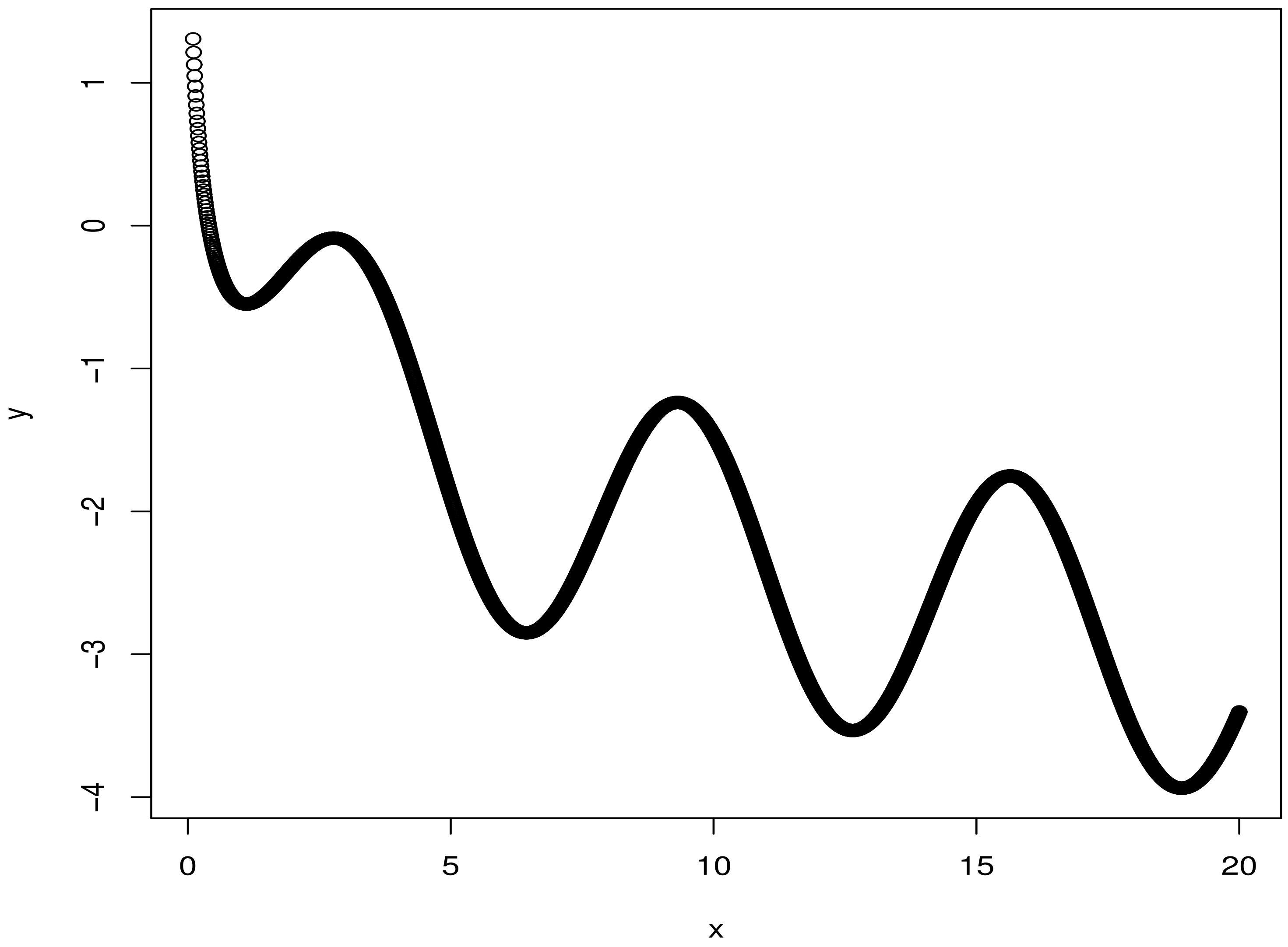
Accordingly, we can use the so called <u>numerical derivative</u> given by

$$f'(x) \approx \frac{f(x+\Delta x) - f(x)}{\Delta x} \tag{6}$$

for small Δx close to zero.

(Exercise) Use $\Delta x = 0.001$ in (6) and apply the numerical derivative method to find the value that maximizes $-x^2$.

A Picture is Worth One Thousand Words $f(x) = -log(x) - cos(x)$



Warnings

1. Numerical methods only produce local optimization, so try a variety of initial values to see if they lead to the same result
2. Check boundary if x is restricted (bounded)
3. If x is restricted, we can use a method called grid search (learned later)
4. There is trouble for the algorithm to converge when Hessian is close to zero
5. If possible, draw a plot of the function

Homework 2 (see syllabus for due date)

For this homework, you need to show me both the R codes and results.

1. (2 points) Use bisection method to solve equation (1) on page 5 (Hint: try $x^{small} = 0, x^{big} = 1$)

2. (3 points) Apply the Newton's method to find the value that maximizes

$$-x^2 + \log(x)$$

   where $\log(x)$ uses e not 10 as base.