Introduction to Numerical Methods (Algorithm)

Example: Find the internal rate of return (IRR)

Consider an investor who pays CF_0 to buy a bond that will pay coupon interest CF_1 after one year and CF_2 (coupon interest plus face value) after two years. The investor wants to find the internal rate of return or yield to maturity that solves the following equation

$$CF_0 = \frac{CF_1}{1+IRR} + \frac{CF_2}{(1+IRR)^2} \tag{1}$$

Letting $x = \frac{1}{1+IRR}$, we can rewrite the polynomial equation as

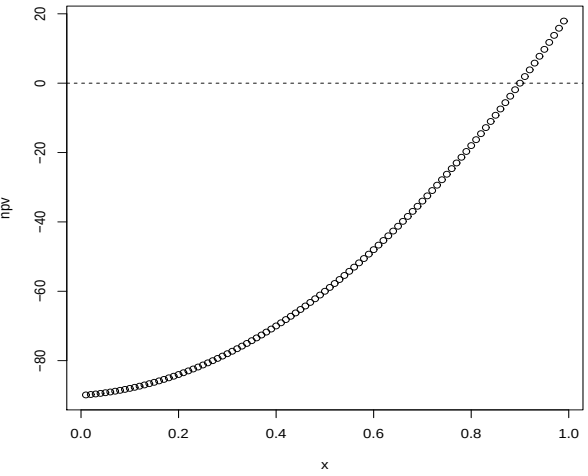
$$0 = -CF_0 + CF_1 x + CF_2 x^2 \tag{2}$$

To fix idea, let $CF_0 = 90, CF_1 = 10, CF_2 = 100$. So we want to solve

$$0 = -90 + 10x + 100x^2 \tag{3}$$

Eye-ball method

We can plot a range of possible values of x against $-90+10x+100x^2$, which is called net present value (npv) in finance. The root is a special x that makes npv equal zero. From the graph below we see $x = 0.9$ is the root, and the resulting IRR is $IRR = \frac{1}{x} - 1 = \frac{1}{0.9} - 1 = 0.11$.



R code

The R codes for the eye-ball method is

```
x = seq(0.01, 0.99, by = 0.01)
npv = rep(0, length(x))
for (i in seq(1,length(x))) {
npv[i] = -90 + 10*x[i] + 100*x[i]^2
}
plot(x, npv)
abline(h=0,lty=2)
cbind(npv, x)
```

While loop

We can use a while loop to automatically find the root without using eyeball.

```
x = 0.01
npv = -90 + 10*x + 100*x^2
while(npv < 0) {
x = x + 0.001
npv = -90 + 10*x + 100*x^2
}
cat("root x is ", x, "\n")
cat("IRR is ", 1/x-1, "\n")
```

Basically, the npv changes from being negative to positive at the root x, where the while loop stops. Does this sound like artificial intelligence?

Bisection method

The fact that npv switches the sign around the root motivates the bisection method, which

1. evaluates npv at two points (a, b) that produce opposite signs of npv
2. lets the tentative root be $\frac{a+b}{2}$, and evaluate both $\left(a, \frac{a+b}{2}\right)$ and $\left(\frac{a+b}{2}, b\right)$
3. picks whichever pair that gives the npv with opposite signs, and let the average of them be the next tentative root
4. the iteration continues until the npv of the tentative root is sufficiently close to zero

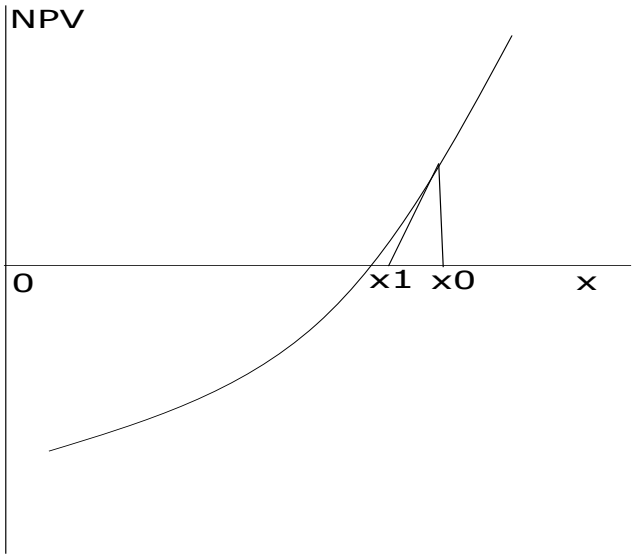
The bisection method is justified by the intermediate value theorem, which states that given two points a and b such that $f(a)$ and $f(b)$ have opposite signs, a continues f must have at least one root in the interval $[a, b]$.

R code

```
npv = function(x) {
y = -90 + 10*x + 100*x^2; return(y)
}
a = 0.10; b = 0.99; root = (a+b)/2
while (npv(a)*npv(b)<0 & abs(npv(root))>0.001) {
root = (a+b)/2
if (npv(a)*npv(root)<0) {a = a; b = root}
else {a = root; b = b}
}
cat("root x is ", root, "\n")
cat("IRR is ", 1/root-1, "\n")
```

Newton–Raphson method

Notice that the npv is a continuous and differentiable (smooth) function of x. So we can pick an initial value x_0, draw a tangent line at $npv(x_0)$ and use the intersection of the tangent line and the horizonal axis as the next value x_1. See the graph below for an illustration. Again, the iteration continues until the npv is very close to zero.



Newton–Raphson method

Mathematically, let $npv'()$ denote the first order derivative, then the first iteration is

$$x_1 = x_0 - \frac{npv(x_0)}{npv'(x_0)} \tag{4}$$

which is based on the definition of slope of tangent line

$$\frac{npv(x_0)}{x_0 - x_1} = npv'(x_0)$$

More generally, after the i-th iteration, the $i+1$-th iteration is

$$x_{i+1} = x_i - k_i \left(\frac{npv(x_i)}{npv'(x_i)} \right) \tag{5}$$

where the step size k_i is used to avoid overshooting

R code

```
npv = function(x) {
y = -90 + 10*x + 100*x^2; return(y)
}
npvprime = function(x) {
y = 10 + 200*x; return(y)
}
root = 0.98
while (abs(npv(root))>0.001) {
step = npv(root)/npvprime(root)
while (abs(step)>0.1) {step = step/10}
root = root- step
}
cat("root x is ", root, "\n")
cat("IRR is ", 1/root-1, "\n")
```

Comparison of three numerical methods

1. The eyeball method works only when we know the possible range of roots. It does not assume continuity or differentiability.
2. The bisection method assumes the function is continuous and we can find two points with opposite signs
3. The Newton–Raphson method assumes differentiability and we can calculate the first order derivative

Comparison of numerical methods and analytical solution

Analytical solution exists for the quadratic equation (2), but are unavailable for a polynomial equation with order greater than two. For those highly non-linear equations, we have to use numerical methods.

Numerical methods for optimization problem

Suppose we want to maximize an objective function $f(x)$ (such as a total utility function). Assuming the objective function is differentiable, a necessary condition is that the first order derivative is equal to zero

$$f'\left(x^{optimal}\right) = 0 \tag{6}$$

The optimal value $x^{optimal}$ is just the root that solves the equation (6). We already know numerical methods can be used to find the roots (and in this case, optimal values). For example, the optimal consumption that maximizes total utility satisfies the condition (equation) that marginal utility is equal to zero

Numerical methods and initial values

The solution for the equation (6) can be either local maximum or local minimum. To tell which is which, and whether the local optimum is a global one, we need to try different initial values and compare the objective values.

Exercises

1. Draw a graph to illustrate the idea behind the bisection method
2. Modify the bisection method code so that the root is not a simple average, but a weighted average of end points $wa + (1-w)b$. How to determine the weight w?
3. Try to remove

   ```
   while (abs(step)>0.1) {step = step/10}
   ```

   from the Newton–Raphson method code, and explain the difference you see in the result. What is overshooting?