Principles of Financial Computing

Yuh-Dauh Lyuu
Dept. Computer Science & Information Engineering
and
Department of Finance
National Taiwan University

References

- Yuh-Dauh Lyuu. *Financial Engineering & Computation: Principles, Mathematics, Algorithms*. Cambridge University Press. 2002.
- Official Web page is

  `www.csie.ntu.edu.tw/~lyuu/finance1.html`

- Check

  `www.csie.ntu.edu.tw/~lyuu/capitals.html`

  for some of the software.

Useful Journals

- *Journal of Computational Finance.*
- *Journal of Derivatives.*
- *Journal of Financial Economics.*
- *Journal of Finance.*
- *Journal of Fixed Income.*
- *Journal of Futures Markets.*
- *Journal of Financial and Quantitative Analysis.*
- *Journal of Real Estate Finance and Economics.*
- *Mathematical Finance.*
- *Review of Financial Studies.*
- *Review of Derivatives Research.*

Introduction

A Very Brief History of Modern Finance

- 1900: Ph.D. thesis *Mathematical Theory of Speculation* of Bachelier (1870–1946).
- 1950s: modern portfolio theory (MPT) of Markowitz.
- 1960s: the Capital Asset Pricing Model (CAPM) of Treynor, Sharpe, Lintner (1916–1984), and Mossin.
- 1960s: the efficient markets hypothesis of Samuelson and Fama.
- 1970s: theory of option pricing of Black (1938–1995) and Scholes.
- 1970s–present: new instruments and pricing methods.

A Very Brief and Biased History of Modern Computers

- 1930s: theory of Gödel (1906–1978), Turing (1912–1954), and Church (1903–1995).
- 1940s: first computers (Z3, ENIAC, etc.) and birth of solid-state transistor (Bell Labs).
- 1950s: Texas Instruments patented integrated circuits; Backus (IBM) invented FORTRAN.
- 1960s: Internet (ARPA) and mainframes (IBM).
- 1970s: relational database (Codd) and PCs (Apple).
- 1980s: IBM PC and Lotus 1-2-3.
- 1990s: Windows 3.1 (Microsoft) and World Wide Web (Berners-Lee)

What This Course Is About

- Financial theories in pricing.
- Mathematical backgrounds.
- Derivative securities.
- Pricing models.
- Efficient algorithms in pricing financial instruments.
- Research problems.

Analysis of Algorithms

Computability and Algorithms

- Algorithms are precise procedures that can be turned into computer programs.
- Uncomputable problems.
- Computable problems.
  - Intractable problems.
  - Tractable problems.

Complexity

- Start with a set of basic operations which will be assumed to take one unit of time.
- The total number of these operations is the total work done by an algorithm (its computational complexity).
- The space complexity is the amount of memory space used by an algorithm.
- Concentrate on the abstract complexity of an algorithm instead of its detailed implementation.
- Complexity a good guide to an algorithm's *actual* running time.

Asymptotics

- Consider the search algorithm on p. 12.
- The worst-case complexity is n comparisons.
- There are operations besides comparison.
- We care only about the asymptotic growth rate not the exact number of operations.
  - For example, the complexity of maintaining the loop is subsumed by the complexity of the body of the loop.
- The complexity is hence $O(n)$.

Algorithm for Searching an Element

```
input:    x, n, A_i  (1 ≤ i ≤ n);
integer   k;
for (k = 1 to n)
          if [x = A_k] return k;
return not-found;
```

Common Complexities

- Linear time if its complexity is $O(N)$.
- Quadratic time if its complexity is $O(N^2)$.
- Cubic time if its complexity is $O(N^3)$.
- Exponential time if its complexity is $O(2^N)$.
- Possible for an exponential-time algorithm to perform well on "typical" inputs.

A Common Misconception about Performance

Is a reduction of the running time from ten seconds to five seconds as significant as that from ten minutes to five minutes?

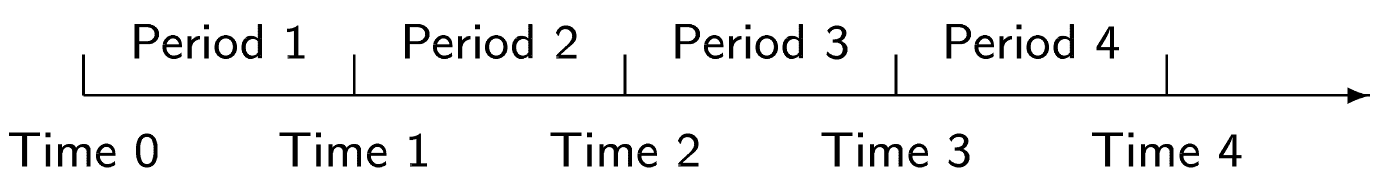
Yes.

A Word on "Recursion"

- In computer science, it means the way of attacking a problem by solving smaller instances of the same problem.
- In finance, "recursion" loosely means "iteration."

Basic Financial Mathematics

The Time Line



Time Value of Money

$$\begin{aligned} \text{FV} &= \text{PV}(1+r)^n, \\ \text{PV} &= \text{FV} \times (1+r)^{-n}. \end{aligned}$$

FV (future value); PV (present value); r: interest rate.

Periodic Compounding

If interest is compounded m times per annum,

$$\mathrm{FV} = \mathrm{PV}\left(1 + \frac{r}{m}\right)^{nm}. \tag{1}$$

Common Compounding Methods

- Annual compounding: $m = 1$.
- Semiannual compounding: $m = 2$.
- Quarterly compounding: $m = 4$.
- Monthly compounding: $m = 12$.
- Weekly compounding: $m = 52$.
- Daily compounding: $m = 365$.

Easy Translations

- An interest rate of r compounded m times a year is equivalent to an interest rate of r/m per $1/m$ year.
- If a loan asks for a return of 1% per month, the annual interest rate will be 12% *with monthly compounding.*

Example

- Annual interest rate is 10% compounded twice per annum.
- Each dollar will grow to be

$$[\,1 + (0.1/2)\,]^2 = 1.1025$$

one year from now.

- The rate is equivalent to an interest rate of 10.25% compounded once *per annum*.

Continuous Compounding

- As $m \to \infty$ and $(1 + \frac{r}{m})^m \to e^r$ in Eq. (1),

$$\text{FV} = \text{PV}e^{rn},$$

where $e = 2.71828\ldots$.

- Continuous compounding is easier to work with.
  - If the annual interest rate is r_1 for n_1 years and r_2 for the following n_2 years, the FV of one dollar will be

$$e^{r_1 n_1 + r_2 n_2}.$$

Efficient Algorithms for PV and FV

- The PV of the cash flow $C_1, C_2, \ldots, C_n$ at times $1, 2, \ldots, n$ is

$$\frac{C_1}{1+y} + \frac{C_2}{(1+y)^2} + \cdots + \frac{C_n}{(1+y)^n}.$$

- Computed by the algorithm on p. 25 in time $O(n)$.

Algorithm for Evaluating PV

```
input:  y, n, C_t  (1 ≤ t ≤ n);
real    x, d;
x := 0;
d := 1 + y;
for (i = n down to 1) {
        x := (x + C_i)/d;
}
return x;
```

The Idea Behind p. 25: Horner's Rule

- This idea is

$$\left(\cdots \left(\left(\frac{C_n}{1+y} + C_{n-1} \right) \frac{1}{1+y} + C_{n-2} \right) \frac{1}{1+y} + \cdots \right) \frac{1}{1+y}.$$

- Due to Horner (1786–1837) in 1819.
- The most efficient possible in terms of the absolute number of arithmetic operations.