# Math Tools: Recursive Methods

Revised: July 8, 2014

The concept of recursion runs throughout modern mathematics and computer science. The same is true of economics. There's a reason the leading PhD textbook in macroeconomics is called *Recursive Macroeconomic Theory*. Such work in economics reflects, in large part, the adoption by economists of methods developed elsewhere.

What follows is a short informal introduction to the idea and a start on the kinds of applications you'll find in economics and finance.

## 1 Recursion with numbers

The idea is to characterize a sequence of items, indexed by an integer $n = 0, 1, 2, \ldots$, by a rule that connects each item to the next one. If we label the items $x_n$, the rule might be expressed

$$x_{n+1} = g(x_n). \tag{1}$$

If we have a starting point, say $x_0$, the rule tells us how to compute as many succeeding items as we wish. We would say that the set $\{x_n\}$ is generated recursively and refer to (1) as the defining recurrence relation. A "solution" to (1) is a formula that expresses $x_n$ as a function of $n$.

Examples:

- Linear difference equation. Let

$$x_{n+1} = ax_n. \tag{2}$$

  This has the solution $x_n = a^n x_0$. It converges to zero if $|a| < 1$, but it's the solution either way.

- Logistic map. Let

$$x_{n+1} = ax_n(1 - x_n)$$

  with $0 < a \leq 4$. If you try some experiments, you'll see that it generates wildly different behavior depending on the value of $a$. You might set $x_0 = 0.3$ and $a = (0.98, 1.5, 2.5, 3.25, 3.5)$, generate (say) 20 terms, and graph the output. See Wikipedia. The point, which we won't develop further, is that even quite simple nonlinear recurrences can generate complex behavior.

- Combinatorics. Legendary computer scientist Herbert Wilf notes that many combinatoric identities satisfy recurrences. For example, the binomial coefficients,

$$f(n,k) \;=\; \left( \begin{array}{c} n \\ k \end{array} \right) \;=\; \frac{n!}{k!(n-k)!},$$

are the solution to

$$f(n,k) \;=\; f(n-1,k) + f(n-1,k-1)$$

starting with $f(n,0) = 1$.

- Fibonacci numbers. The Fibonacci numbers are generated by the second order system

$$f_{n+1} \;=\; f_n + f_{n-1}$$

starting with $f_0 = 0$ and $f_1 = 1$. In matrix terms, we can write this as $x_{n+1} = Ax_n$ with

$$x_n \;=\; \left[ \begin{array}{c} f_n \\ f_{n-1} \end{array} \right], \quad A \;=\; \left[ \begin{array}{cc} 1 & 1 \\ 1 & 0 \end{array} \right].$$

This is similar to (2), but here $x_n$ is a vector. The matrix $A$ has eigenvalues $(\lambda_1, \lambda_2)$ satisfying $\lambda^2 - \lambda - 1 = 0$. The "solution" has the form $f_n = c_1\lambda_1^n + c_2\lambda_2^n$ for constants $(c_1, c_2)$ that satisfy the initial conditions.

This is a common example in computer science courses. A recursive version of a Matlab program to compute Fibonacci numbers is

```
function answer = f(n)

if n==0
    answer = 0;
elseif (n==1)
    answer = 1;
else
    answer = f(n-1)+f(n-2);
end

end
```

Note that the function `f` refers to itself — it's recursive in the sense the word us used in computer science. In Matlab, we would save this as a file called `f.m`, then call it by typing (say) `f(8)` in the command line or as a line in another program. (If you enter a fraction, it blows up, so a better function would check and generate an error message.)

- Mean and variance. John Cook describes the Welford method of computing the mean and variance recursively. Consider a sequence of observations: $x_1, x_2, \ldots$. We can compute rolling estimates of the mean and variance from

$$\begin{aligned} M_n &= M_{n-1} + (x_n - M_{n-1})/n \\ S_n &= S_{n-1} + (x_n - M_{n-1}) * (x_n - M_n) \end{aligned}$$

starting with $M_1 = x_1$ and $S_1 = 0$. Do a few terms to assure yourself that $M_n$ is the mean of the first $n$ observations and $S_n$ is the sum of the squared deviations from the mean. The standard estimator of the variance $s^2$ is therefore $S_n/(n-1)$ (although I prefer to divide by $n$, always).

- Natural numbers. The natural numbers are the set $\mathbb{N} = \{0, 1, 2, \ldots\}$. We can define them recursively with the rules: (i) 0 is in $\mathbb{N}$ and (ii) if $n$ is in $\mathbb{N}$ then so is $n + 1$.

  Which reminds me of an old George Gamov story. In the story, the Hilbert Hotel has an infinite number of rooms numbered $1, 2, 3, \ldots$. By law, it must save one for the King, but the innkeeper fills them all anyway. When asked, he says: "No problem, I can always get an open room by asking everyone to move over one." And if we move the person in room $j$ to room $2j$, we open up an infinite (countable) number of rooms, all the odd-numbered ones.

## 2 Markov processes

?? ***** Explain markov structure, everything is a function of state $z_t$.

Put this below??

## 3 Recursion with functions

In economics and finance we often run across recursion with functions. Suppose we have a sequence of functions $f_n(x)$ over some domain of $x$ defined by

$$f_{n+1}(x) \quad = \quad g[f_n(x)]$$

for some $g$. Or even better, suppose the recurrence has a fixed point:

$$f(x) \quad = \quad g[f(x)].$$

We usually drop the $x$ and write $f = g(f)$. We're talking about a more complex object here — a function $f$ rather than a number $x$ — but the idea is similar to (1).

The examples below illustrate the idea, but we skip the technical parts: we do not show that a solution exists or is unique. In applications that's sometimes obvious, sometimes not.

**Stochastic processes.** ???

**Forecasting.** ???

**Bond pricing.** You may recall that modern asset pricing is based on the no-arbitrage theorem: there exists a positive pricing kernel $m$ that satisfies $E(mr) = 1$ for returns $r$ on all assets. This takes a particularly elegant form with bonds. The basic idea should be familiar from our earlier work, but to be concrete:

- Consider a stationary Markov environment with state variable $x_t$ and conditional probability density $p(x_{1+1}|x_t)$.
- Asset pricing follows from the no-arbitrage theorem,

$$E_t\big[m(x_t, x_{t+1})r(x_t, x_{t+1})\big] \quad = \quad 1.$$

  Here $m(x_t, x_{t+1}) > 0$ is a pricing kernel, $r(x_t, x_{t+1})$ is the (gross) return on an arbitrary asset, and $E_t$ is the expectation conditional on the current state $x_t$ — the expectation computed from $p(x_{t+1}|x_t)$, in other words.
- A bond of maturity $n$ is a claim to a payment of one in $n$ periods. We find bond prices recursively, starting with

$$q^1(x_t) \quad = \quad E_t\big[m(x_t, x_{t+1})\big];$$

  that is, the bond price is a function of the state $x_t$. Bonds of longer maturity follow from the recursion

$$q^{n+1}(x_t) \quad = \quad E_t\big[m(x_t, x_{t+1})q^n(x_{t+1})\big]. \tag{3}$$

  In words: an $n+1$-period bond is a claim to an $n$-period bond in one period. This works for $n = 0$ as well with $q^0(x_t) = 1$.

We studied examples earlier in which $x_t$ was one-dimensional and the bond price was log-linear: $\log q^n(x_t) = A_n + B_n x_t$. The recursions in the bond price functions in these models translate into recursions in the coefficients $(A_n, B_n)$.

Another class of tractable models is based on finite-state Markov chains: $x_t$ takes on a finite number of values, say $i = 1, \ldots, I$. Then the function $q^n(x_t)$ is a vector, say $q^n$, with one element for every state. By tradition, we collect the conditional ("transition") probabilities in a matrix $P = [p_{ij}]$, where $p_{ij}$ is the probability that $x_{t+1} = j$ given that $x_t = i$. Similarly, the pricing kernel is a matrix of elements $[m_{ij}]$.

In this setting, a one-period bond price in state $i$ is

$$q_i^1 \quad = \quad \sum_j p_{ij} m_{ij}.$$

If we define the matrix $B = [b_{ij}] = [p_{ij} m_{ij}]$, then in matrix terms, the one-period bond price function/vector is

$$q^1 \quad = \quad Bq^0,$$

where $q^0$ is a vector of ones — the prices of 0-maturity bonds, namely one. Other bond prices follow recursively:

$$q^{n+1} \quad = \quad Bq^n.$$

You should convince yourself that this is (3) applied to this environment. If we substitute, we see that $q^n = B^n q^0$, so anything we know about powers of positive matrices can be put to work.

**Equity pricing.** A dividend paying stock is a more complicated object. In the same environment as before, let the dividend in state $x_t$ be $d(x_t)$. The ex-dividend value of a share might be expressed recursively as

$$v(x_t) = E_t\big\{m(x_t, x_{t+1})[d(x_{t+1}) + v(x_{t+1})]\big\}. \tag{4}$$

In words: equity today is a claim to two things tomorrow, a dividend and the same share of equity.

Note that this is a functional equation: the unknown is the function $v$. It's also recursive: you need to know $v$ on the right to compute $v$ on the left. You're now as ready as you'll ever be to understand recursion jokes. "To understand recursion, you need to understand recursion." Or Google "recursion." You get back: "Did you mean: recursion?"

One way to think about this is as the limit of a finite horizon. Suppose we value next period's dividend. We might express its value by

$$v^1(x_t) = E_t\big\{m(x_t, x_{t+1})d(x_{t+1})\big\}.$$

The superscript 1 here means we're valuing one period of dividends. We can value two periods of dividends recursively with

$$v^2(x_t) = E_t\big\{m(x_t, x_{t+1})[d(x_{t+1}) + v^1(x_{t+1})]\big\}.$$

In general, we can value $n+1$ periods of dividends with the recursion

$$v^{n+1}(x_t) = E_t\big\{m(x_t, x_{t+1})[d(x_{t+1}) + v^n(x_{t+1})]\big\},$$

starting with $v^0(x_t) = 0$ (the value of zero dividends is zero). As we increase $n$, we have more and more dividends. We might imagine, if all goes well, that as $n$ gets larger and larger, we approach (4).

Unlike bond pricing, loglinear examples don't work. There are some popular loglinear approximations, but that's too much work for now. We can, however, adapt the Markov chain setup we used with bonds. Suppose the dividend is a vector $d$, with one element for each state. Then the pricing relation (4) becomes

$$v = B(d + v).$$

The solution is $v = (I - B)^{-1}Bd$. Alternatively, we can substitute repeatedly to get $v = Bd + B^2d + B^3d + \cdots$, the present discounted value version of the equity price.

**Perpetual options.** Consider the option to buy one share of stock next period for strike price $k$. The value today in state $x_t$ is

$$q(x_t) = E_t\big\{m(x_t, x_{t+1})[v(x_{t+1}) - k]^+\big\},$$

where $x^+ = \max\{0, x\}$. Evidently we exercise in states where $v(x_{t+1}) - k$ is positive and not in other states.

A perpetual option allows us to wait: if we don't exercise now we can hold the option for another period, and do this again, forever. Valuation has a recursive form:

$$q(x_t) \quad = \quad \max\left\{v(x_t) - k, E_t\left[m(x_t, x_{t+1})q(x_{t+1}\right]\right\}.$$

That is: we either exercise now and get $v(x_t) - k$ (the first branch of the max) or continue to hold the option and get the discounted value of the option next period (the second branch).

**Utility.** In dynamic settings, we often use the additive utility function,

$$U(x_t) \quad = \quad E_t \sum_{j=0}^{\infty} \beta^j u[c(x_{t+j})],$$

for some function $u$ and discount factor $0 < \beta < 1$. The familiar power utility consists of $u(c) = c^{1-\alpha}/(1-\alpha)$ with $\alpha > 0$. Here $U_t$ means "utility from date $t$ on." We can rewrite it as

$$U(x_t) \quad = \quad u[c(x_t)] + \beta E_t[U(x_{t+1})].$$

Why does this work? Because $U(x_{t+1})$ takes care of the rest of the terms.

As in our other examples, we can envision building this up term by term,

$$U^{n+1}(x_t) \quad = \quad u[c(x_t)] + \beta E_t[U^n(x_{t+1})],$$

starting with $U^0(x_t) = 0$. If all goes well, the unknown function $U(x_t)$ is the limit of the $U^n(x_t)$'s.

## 4 Dynamic optimization: deterministic problems

Canonical problem, Lagrangian and recursive methods. Show they're the same. Take limit of finite horizon.

Examples. LQ, Brock-Mirman.

## 5 Dynamic optimization: stochastic problems

Recursive approach with an $E$.

### Practice problems

1. *Discounted cash flows.* Our goal here is to simplify the pricing relation (4) and derive a more conventional valuation of equity as the expected discounted value of future dividends.
   (a) Simplify (4) using $m(x_t, x_{t+1}) = \delta$ and replacing dependence on the state $x_t$ with a subscript $t$ — that is, by replacing $v(x_t)$ with $v_t$.

(b) Use your simplification to express equity's value as

$$v_t = \sum_{j=1}^{n} \delta^j E_t(d_{t+j}) + \delta^n E_t(v_{t+n}).$$

(c) What happens as $n$ gets large? What happens to the second term on the right above?

(d) What does this example leave out that's present in (4)?

2. *Consols.* A consol is a perpetual bond: it pays a coupon $c$ every period forever. How would we value one in an exponential-affine setting? Consider the pricing kernel

$$\begin{aligned} \log m_{t+1} &= \delta + x_t + \lambda w_{t+1} \\ x_{t+1} &= \varphi x_t + \sigma w_{t+1}, \end{aligned}$$

where $0 < \varphi < 1$ and $\{w_t\}$ is a sequence of independent standard normal random variables. As we've seen, this structure gives us loglinear bond prices: $\log q^n(x_t) = A_n + B_n x_t$.

(a) Use (3) to find recursions for the coefficients $(A_n, B_n)$.

(b) Use your answer to find the price of a consol.

*Comment about notation*: $m(x_t, x_{t+1})$ is the most accurate notation, but we (meaning economists in general) often use $m_{t,t+1}$ or $m_{t+1}$ to mean the same. They're less accurate, but easier to write.

3. *Representative agent asset pricing.* (Adapted from an exercise in Ljungqvist and Sargent, *Recursive Macroeconomic Theory*.) Consider the representative agent model with additive preferences

$$U_t = E_t \sum_{n=0}^{\infty} \beta^n c_{t+n}^{1-\alpha}/(1-\alpha).$$

The agent's marginal rate of substitution is the pricing kernel, $m_{t+1} = \beta(c_{t+1}/c_t)^{-\alpha}$.

Now suppose the growth rate $g_{t+1} = c_{t+1}/c_t$ follows

$$\begin{aligned} \log g_{t+1} &= x_t + \theta w_{t+1} \\ x_{t+1} &= \varphi x_t + \sigma w_{t+1} \end{aligned}$$

with $0 < \varphi < 1$ and $\{w_t\}$ a sequence of iid standard normal random variables.

(a) Bond prices in this economy take the form $\log q^n(x_t) = A_n + B_n x_t$. Use the pricing relation $q_t^{n+1} = E_t(m_{t+1} q_{t+1}^n)$ to compute recursions for $(A_n, B_n)$.

(b) Show that future consumption has the form: $c_{t+1} = c_t g_{t+1}$, $c_{t+2} = c_t g_{t+1} g_{t+2}$, $c_{t+n} = c_t g_{t+1} \cdots g_{t+n}$.

(c) Use this to express the $n$th term in the utility function, namely $E_t c_{t+n}^{1-\alpha}/(1-\alpha)$, in terms of growth rates. Compute these terms recursively for $n = 0, 1, 2, \ldots$. Hint: you might try $\log(E_t c_{t+n}^{1-\alpha}) = F_n + G_n c_t + H_n x_t$.

(d) Express utility $U_t$ as a function of current consumption $c_t$ and the state variable $x_t$.

(e) Now consider equity, a claim to future dividends, $d_{t+n} = c_{t+n}^{\lambda}$ for $n = 0, 1, 2, \ldots$. Show how you can value dividends, one at a time, recursively.

4. Bond pricing in a Markov chain. Coupon bonds? Perpetuity?

5. Perpetual option. Markov chain version.

6. *Perpetual option: BSM version.* With Black-Scholes-Merten structure, the price of a perpetual option can be solved by hand. See the appendix.