week15-01–gen-functions

April 14, 2025

1 George McNinch Math 87 - Spring 2025

2 Week 14

3 Recurrence relations & generating functions

4 Discrete-time problems

Our goal in this notebook is to discuss some deterministic models which are naturally "discrete in the time parameter". We'll try to make that precise as we go forward!

4.1 Example

Suppose that you have a \$2000 balance on your credit card. Each month, you charge \$500 and pay \$750 off of the balance, but interest is charged at a rate of 1.5% of the unpaid balance each month.

Write b_i for the balance after i months. Thus $b_0 = 2000$, and the relation

$$(\clubsuit) \quad b_{i+1} = b_i + 500 - 750 + 0.015 b_i = 1.015 b_i - 250$$

holds.

Question: How long does it take to pay off the balance?

The description ($\clubsuit$) is known as *a recurrence relation.*

In order to give a `code`-based solution, we can use recursion, as follows:

[1]:
```python
def balance(i):
    ## return the balance after i months
    if i == 0:
        return 2000
    else:
        return (1.015)*balance(i-1) -250

def month_range(n):
    res = [f"{i:02d} - $ {balance(i):.02f}" for i in range(n)]
    return "\n".join(res)
```

```
print(month_range(10))
```

```
00 - $ 2000.00
01 - $ 1780.00
02 - $ 1556.70
03 - $ 1330.05
04 - $ 1100.00
05 - $ 866.50
06 - $ 629.50
07 - $ 388.94
08 - $ 144.78
09 - $ -103.05
```

Thus the loan is repaid after 9 months….

But this required us to *guess* a possible solution and use the code to check whether it works.

4.2 Example

Suppose that you want to buy a house. You will need to take out a \$300,000 mortgage to do so. The interest accumulates at a rate of 0.4 % per month (4.8% per year), and your monthly payment is \$1600.

How long does it take to pay off the mortgage?

Again, if b_i is the balance after i months, then $b_0 = 300000$ and

$$b_{i+1} = b_i + 0.004b_i - 1600 = 1.004b_i - 1600$$

This is pretty similar to the previous example….

4.3 Example

Consider an idealized rabbit population. Assume that a newly born pair of rabbits, one male and one female, are put in a field. Rabbits are able to mate at the age of one month, so that at the end of her second month, a female can produce a new pair of rabbits. A mating pair always produces a new mating pair (one male and one female) every month from the second month on. For purposes of this model, we are going to ignore rabbit mortality (we'll assume that rabbits don't die…)!!

How many pairs of rabbits will there be after k months?

Write f_k = number of pairs of rabbits in the field after k months. Thus $f_0 = 0$ and $f_1 = 1$.

For $k \geq 2$ all pairs of rabbits who are at least two months old reproduce, and all pairs from month $k - 1$ are still alive, so:

$$f_k = f_{k-1} + f_{k-2}$$

This is the Fibonacci sequence

```
[2]: def fibonacci(n):
         if n==0:
             return 0
         if n==1:
             return 1
         else:
             return fibonacci(n-1) + fibonacci(n-2)

     fib = [f"{m} - {fibonacci(m)}" for m in range(10)]
     print("\n".join(fib))
     print()

     fib_by_5 = [f"{m} - {fibonacci(m)}" for m in range(10,40,5)]
     print("\n".join(fib_by_5))
```

```
0 - 0
1 - 1
2 - 1
3 - 2
4 - 3
5 - 5
6 - 8
7 - 13
8 - 21
9 - 34

10 - 55
15 - 610
20 - 6765
25 - 75025
30 - 832040
35 - 9227465
```

4.4 Generating functions

It is often desirable to have a *closed-form* description of a solution to a recurrence problem. For *some* problems involving recurrence relations, we can find a so-called *generating function* which provides a nice description of the solution.

Our main trick is going to be usage of a so-called *formal power series*:

$$f(x) = \sum_{k=0}^{\infty} f_k x^k$$

where we still need to specify the *coefficients* f_k of $f(x)$.

Let's consider such a formal series in the case of the Fibonacci example. Thus the coefficient f_k of x^k in the series $f(x)$ is precisely the k-th Fibonacci number.

This means that the coefficients satisfy the recurrence relation

$$f_{k+2} = f_{k+1} + f_k$$

for $k \geq 0$.

We are going to argue that this recurrence relationship leads to an algebraic identity involving the formal power series $f(x)$.

Multiplying each side of this equality by x^{k+2} we obtain

$$f_{k+2}x^{k+2} = f_{k+1}x^{k+2} + f_k x^{k+2}.$$

Now, summing over all $k \geq 0$ and *ignoring convergence issues* (!) we obtain

$$(\clubsuit) \quad \sum_{k=0}^{\infty} f_{k+2}x^{k+2} = \sum_{k=0}^{\infty} f_{k+1}x^{k+2} + \sum_{k=0}^{\infty} f_k x^{k+2}$$

Let's notice that

$$(\mathbf{a}) \quad \sum_{k=0}^{\infty} f_k x^{k+2} = x^2 \sum_{k=0}^{\infty} f_k x^k = x^2 f(x)$$

Moreover, reindexing the left-most term in ($\clubsuit$) via $j = k+2$ we get

$$(\mathbf{b}) \quad \sum_{k=0}^{\infty} f_{k+2}x^{k+2} = \sum_{j=2}^{\infty} f_j x^j = f(x) - f_0x^0 - f_1x^1 = f(x) - f_0 - f_1x$$

and

$$(\mathbf{c}) \quad \sum_{k=0}^{\infty} f_{k+1}x^{k+2} = x\sum_{k=0}^{\infty} f_{k+1}x^{k+1} = x\sum_{k=1}^{\infty} f_k x^k = x(f(x) - f_0)$$

Combining $(\mathbf{a}), (\mathbf{b})$ and $(\mathbf{c})$ with the recurrence relation ($\clubsuit$), we find that

$$f(x) - f_0 - f_1x = x(f(x) - f_0) + x^2 f(x)$$

Thus

$$f(x) - xf(x) - x^2 f(x) = f_0 + f_1x - f_0x$$

$\implies$

$$(1 - x - x^2)f(x) = f_0 + (f_1 - f_0)x$$

$\Longrightarrow$

$$f(x) = \frac{f_0 + (f_1 - f_0)x}{1 - x - x^2}$$

Since $f_0 = 0$ and $f_1 = 1$, we find an *identity of formal power series*:

$$f(x) = \frac{x}{1 - x - x^2}$$

4.5 Application

The idea is that we can use the Taylor series development at $x = 0$ of $\frac{x}{1 - x - x^2}$ to find a formulat for the coefficients f_k.

Our main tool will be the geometric series:

$$\frac{1}{1 + \alpha x} = 1 - \alpha x + \alpha^2 x^2 - \cdots = \sum_{i=0}^{\infty} (-1)^i \alpha^i x^i$$

We start by factoring $1 - x - x^2 = -(x^2 + x - 1)$; using the quadratic formula, we see that the roots are

$$\phi = \frac{-1 + \sqrt{5}}{2} \qquad \text{and} \qquad \psi = \frac{-1 - \sqrt{5}}{2}.$$

This leads to the factorization:

$$1 - x - x^2 = -(x - \phi)(x - \psi)$$

and we see that these roots satisfy the identities

$$\phi \cdot \psi = -1 \qquad \text{and} \qquad \phi + \psi = -1.$$

Thus

$$1 - x - x^2 = -(x - \phi)(x - \psi) = \phi\psi(x - \phi)(x - \psi) = (\phi x + 1)(\psi x + 1)$$

Let's notice the following identities:

$$(\diamondsuit) \qquad \phi - \psi = \sqrt{5}$$

We now use the method of *partial fractions* to rewrite the series $f(x)$ we must find constants A and B which make the following expression valid:

$$f(x) = \frac{x}{1-x-x^2} = \frac{x}{(1+\phi x)(1+\psi x)} = \frac{A}{1+\phi x} + \frac{B}{1+\psi x}$$

Getting a common denominator on the RHS, we see that A, B are determined by the equation:

$$x = A(1+\psi x) + B(1+\phi x) = (A+B) + (A\psi + B\phi)x$$

So we must have

$$\begin{matrix} A + B = 0 \\ A\psi + B\phi = 1 \end{matrix}; \qquad \text{i.e.} \qquad \begin{bmatrix} 1 & 1 \\ \psi & \phi \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

```
[ ]:
```

Performing row-operations on the corresponding augmented matrix, we see that

$$\begin{bmatrix} 1 & 1 & 0 \\ \psi & \phi & 1 \end{bmatrix} \approx \begin{bmatrix} 1 & 1 & 0 \\ 0 & \phi-\psi & 1 \end{bmatrix} \approx \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1/(\phi-\psi) \end{bmatrix} \approx \begin{bmatrix} 1 & 0 & -1/(\phi-\psi) \\ 0 & 1 & 1/(\phi-\psi) \end{bmatrix}.$$

Combined with ($\diamondsuit$), this shows the following:

$$A = -\frac{1}{\phi-\psi} = \frac{-1}{\sqrt{5}} \qquad \text{and} \qquad B = \frac{1}{\phi-\psi} = \frac{1}{\sqrt{5}}$$

Conclude that

$$f(x) = \frac{x}{1-x-x^2} = \frac{-1}{\sqrt{5}} \cdot \frac{1}{1+\phi x} + \frac{1}{\sqrt{5}} \cdot \frac{1}{1+\psi x}$$

$$= \frac{-1}{\sqrt{5}} \cdot \sum_{i=0}^{\infty} (-1)^i \phi^i x^i + \frac{1}{\sqrt{5}} \cdot \sum_{i=0}^{\infty} (-1)^i \psi^i x^i$$

$$= \sum_{i=0}^{\infty} \left(\frac{(-1)^{i+1}\phi^i + (-1)^i \psi^i}{\sqrt{5}} \right) x^i$$

$$= \sum_{i=0}^{\infty} \left(\frac{(-\psi)^i - (-\phi)^i}{\sqrt{5}} \right) x^i.$$

This leads to the following formula for the Fibonacci numbers:

$$f_i = \frac{(-\psi)^i - (-\phi)^i}{\sqrt{5}}$$

We can write code to implement this, as follows. This code avoids recursion, but gets some noise from the floating point calculations:

[3]:
```
import numpy as np

def fib_closed_form(i):
    m_phi = (-1)*(-1 + np.sqrt(5))/2
    m_psi = (-1)*(-1 - np.sqrt(5))/2
    return (m_psi**i - m_phi**i)/np.sqrt(5)

fib_closed = [f"{m} - {fib_closed_form(m)}" for m in range(10)]
print("\n".join(fib_closed))
print()

fib_closed_by_5 = [f"{m} - {fib_closed_form(m)}" for m in range(10,40,5)]
print("\n".join(fib_closed_by_5))
```

```
0 - 0.0
1 - 1.0
2 - 1.0
3 - 2.0
4 - 3.0000000000000004
5 - 5.000000000000001
6 - 8.000000000000002
7 - 13.000000000000002
8 - 21.000000000000004
9 - 34.00000000000001

10 - 55.000000000000014
15 - 610.0000000000003
20 - 6765.000000000005
25 - 75025.00000000006
30 - 832040.0000000008
35 - 9227465.000000011
```

4.6 Linear Homogeneous recurrence relations

The technique just described for giving a formula for the Fibonacci numbers works more generally for linear homogeneous recurrence relations with constant coefficients.

Let's give a rough formulation of this more general setting; you can see a more details on the definition here. We consider a sequence of quantities $b_0, b_1, b_2, \cdots$

We consider such recurrence relations of order $k \in \mathbb{Z}_{\geq 1}$. This means that there are some coefficients $c_1, c_2, \cdots, c_k$ with $c_k \neq 0$ and a relation for each $n \geq k$ of the form

$$(\clubsuit) \quad b_n = c_1 b_{n-1} + c_2 b_{n-2} + \cdots + c_k b_{n-k}$$

In particular, the Fibonacci sequence is determined by a recurrence relation as above of order $k = 2$.

Note in particular, there is no constant term in the formula for b_i (this is what is meant by the term "homogeneous"). Becaues of this, the credit card and mortgage examples are *not* linear homogeneous recurrence relations (see below, though!)

In the situation of a constant coefficient linear homogeneous recurrence as above, again consider the generating function

$$b(x) = \sum_{i=0}^{\infty} b_i x^i.$$

The expression (♣) can be used to show that the formal power series $b(x)$ identifies with the Taylor expansion of a *rational function of* x – see generating functions. Now one uses the method of partial fractions as we did in the case of the generating function for the Fibonacci numbers.

5 Inhomogenous case?

How should we proceed in the case of an inhomogeneous recurrence relation, like that found in our credit card example and mortgage example??

Let's discuss first the mortgage example. Recall that $b_0 = 300000$ and

$$b_{i+1} = 1.004 b_i - 1600 \quad \text{for} \quad i \geq 0.$$

5.1 First approach

Let's notice that

$$b_{i+1} = 1.004 b_i - 1600 \implies b_{i+1} - 1.004 b_i = -1600$$

and

$$b_{i+2} = 1.004 b_{i+1} - 1600 \implies b_{i+2} - 1.004 b_{i+1} = -1600$$

Subtracting the equations gives

$$b_{i+2} - 2.004 \cdot b_{i+1} + 1.004 \cdot b_i = 0 \implies b_{i+2} = 2.004 \cdot b_{i+1} - 1.004 \cdot b_i$$

Thus we have replaced the inhomogeneous recurrence relation with a homogeneous relation which we can now solve as in the Fibonacci example.

5.2 Second approach

Let's first find a *steady-state* solution. Under what circumstances is it true that $b_i = b_{i+1} = b^*$ for all sufficiently large i?

Well, this equation implies that

$$b^* = 1.004 b^* - 1600 \implies 0.004 b^* = 1600 \implies b^* = 400000$$

This indicates that if the loan had a balance of \$400,000, the loan would never be paid off (in fact, the balance would remain constant!).

However, our assumption was that the loan value *started* at \$300,000. Nevertheless, we can use our steady-state solution $b^* = 400000$ as follows:

Let $c_i = b_i - b^*$. Since

$$b_{i+1} = 1.004b_i - 1600 \qquad \text{and} \qquad b^* = 1.004b^* - 1600$$

we find that

$$b_{i+1} - b^* = 1.004(b_i - b^*)$$

for $i \geq 0$. But this means

$$c_{i+1} = 1.004c_i \qquad \text{for } i \geq 0$$

and it is then easy to see that

$$c_i = (1.004)^i c_0 \qquad \text{for } i \geq 0$$

This means that the generating function for the c_i satisfies

$$c(x) = \sum_{i=0}^{\infty} c_i x^i = \sum_{i=0}^{\infty} (1.004)^i c_0 x^i = \frac{c_0}{1 - 1.004x}$$

Returning to the coefficients b_i, we find now:

$$b(x) = \sum_{i=0}^{\infty} b_i x^i = \sum_{i=0}^{\infty} (c_i + b^*) x^i = \sum_{i=0}^{\infty} c_i x^i + \sum_{i=0}^{\infty} b^* x^i$$

$$= \frac{c_0}{1 - 1.004x} + \frac{b^*}{1 - x}$$

In particular,

$$b_i = c_i + b^* = (1.004)^i c_0 + b^* = (1.004)^i (b_0 - b^*) + b^*$$

Since $b_0 = 300000$ and $b^* = 400000$ we see that

$$b_i = 400000 - 100000(1.004)^i$$

Now,

$$b_i = 0 \implies 4 = 1.004^i \implies \ln(4) = i \cdot \ln(1.004) \implies i = \frac{\ln(4)}{\ln(1.004)}$$

```
[4]: np.log(4)/np.log(1.004)
```

```
[4]: 347.2662762842735
```

```
[5]: 348/12.
```

```
[5]: 29.0
```

Thus the loan is paid off in 348 months, i.e. in 29 years.