A look at Fibonacci through linear algebra

Andrés Casillas García de Presno

July 2025

1 Framework

Let $F_0 = F_1 = 0$, and consider the following recursion

$$F_{n+2} = F_{n+1} + F_n \tag{1}$$

$\forall n \in \mathbb{N}$.

The sequence that arises -known as the Fibonacci sequence- has proven to be relevant in several areas of mathematics. In this article we analyze it using the tools of linear algebra, getting a closed form of the $n-th$ Fibonacci number, namely

$$F_n = \frac{\varphi^n - (-\varphi)^{-n}}{\sqrt{5}}$$

for all $n \in \mathbb{N}$, and showing that

$$\lim_{n\to\infty} \frac{F_{n+2}}{F_{n+1}} = \varphi$$

where $\varphi = \frac{1+\sqrt{5}}{2}$ is the golden ratio.

2 Derivation

One can restate Fibonacci's recurrence as the following linear system:

$$\begin{bmatrix} F_{n+2} \\ F_{n+1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} F_{n+1} \\ F_n \end{bmatrix}$$

where

$$T := \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

In order to compute $\lim_{n\to\infty} \frac{F_{n+2}}{F_{n+1}}$, it suffices to compute T^n for all $n \in \mathbb{N}$, and then take the limit of its entries. We do so by diagonalizing the matrix according to the following

Theorem 1 (Diagonalizable Matrix Theorem) *A matrix $T \in \mathbb{R}^{n\times n}$ is diagonalizable if there exists an invertible matrix P such that*

$$T = PDP^{-1},$$

where D is a diagonal matrix containing the eigenvalues of T.

We now proceed to calculate the eigenvalues and eigenvectors of $T = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$

Eigenvalues of T

$$T = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

The characteristic polynomial is

$$\det(T - \lambda I) = \lambda^2 - \lambda - 1 = 0, \tag{2}$$

with solutions

$$\lambda_1 = \frac{1+\sqrt{5}}{2} = \varphi, \quad \lambda_2 = \frac{1-\sqrt{5}}{2} = (-\varphi)^{-1}.$$

Associated eigenvectors
For λ_1:

$$(T - \lambda_1 I)\mathbf{v}_1 = 0 \implies \mathbf{v}_1 = \begin{bmatrix} \lambda_1 \\ 1 \end{bmatrix}.$$

For λ_2:

$$(T - \lambda_2 I)\mathbf{v}_2 = 0 \implies \mathbf{v}_2 = \begin{bmatrix} \lambda_2 \\ 1 \end{bmatrix}.$$

Change of basis matrix P and its inverse P^{-1}

$$P = \begin{bmatrix} \lambda_1 & \lambda_2 \\ 1 & 1 \end{bmatrix}, \quad P^{-1} = \frac{1}{\lambda_1 - \lambda_2} \begin{bmatrix} 1 & -\lambda_2 \\ -1 & \lambda_1 \end{bmatrix}, \quad D = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix},$$

We can then easily compute T^n by calculating the matrix product

$$T^n = PD^nP^{-1} = \frac{1}{\sqrt{5}} \begin{bmatrix} \lambda_1 & \lambda_2 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \lambda_1^n & 0 \\ 0 & \lambda_2^n \end{bmatrix} \begin{bmatrix} 1 & -\lambda_2 \\ -1 & \lambda_1 \end{bmatrix}$$

and thus

$$\begin{bmatrix} F_{n+1} \\ F_n \end{bmatrix} = PD^nP^{-1} \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

By explicit computation. we get

$$T^n = PD^nP^{-1} = \frac{1}{\sqrt{5}}\begin{bmatrix}\lambda_1^{n+1}-\lambda_2^{n+1} & -\lambda_1\lambda_2^n+\lambda_2\lambda_1^n \\ \lambda_1^n-\lambda_2^n & -\lambda_2\lambda_1^n+\lambda_1\lambda_2^n\end{bmatrix}$$
$$= \frac{1}{\sqrt{5}}\begin{bmatrix}\varphi^{n+1}-(-\varphi)^{-(n+1)} & -\varphi(-\varphi)^{-n}+(-\varphi)^{-1}\varphi^n \\ \varphi^n-(-\varphi)^{-n} & -(-\varphi)^{-1}\varphi^n+\varphi(-\varphi)^{-n}\end{bmatrix}$$

so finally

$$\begin{bmatrix}F_{n+1}\\ F_n\end{bmatrix} = T^n\begin{bmatrix}1\\0\end{bmatrix} = \frac{1}{\sqrt{5}}\begin{bmatrix}\varphi^{n+1}-(-\varphi)^{-(n+1)} & -\varphi(-\varphi)^{-n}+(-\varphi)^{-1}\varphi^n \\ \varphi^n-(-\varphi)^{-n} & -(-\varphi)^{-1}\varphi^n+\varphi(-\varphi)^{-n}\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix}$$

which yields Binet's formula

$$F_n = \frac{\varphi^n-(-\varphi)^{-n}}{\sqrt{5}}$$

for all $n \in \mathbb{N}$.

From this and the fact that $\lim_{n\to\infty}(-\varphi)^{-n} = 0$, it is immediate that

$$\lim_{n\to\infty}\frac{F_{n+2}}{F_{n+1}} = \varphi$$

3 A problem and an elegant solution

It never seizes to amaze me that $\frac{\varphi^n-(-\varphi)^{-n}}{\sqrt{5}}$ is an integer for all natural numbers (since they are defined by the Fibonacci recursion, with base case in the integers). But how do we actually compute this? We still have to calculate φ^n and $(-\varphi)^{-n}$, which don't seem easy since both φ and $(-\varphi)^{-1}$ are irrational numbers. Let's take a closer look at Binet's formula.

Recall $F_n = \frac{\varphi^n-(-\varphi)^{-n}}{\sqrt{5}}$. Note that $\varphi > 1$ -since $\sqrt{5} > 2$, and thus one is tempted to say that $F_n \approx \frac{\varphi^n}{\sqrt{5}}$. This leaves us with a somewhat bad taste, since we just computed an exact formula and now we are going back to approximations. However, I claim that this approximation can get us a better exact formula, if we look at it closely.

We can ask ourselves: how far from F_n is $\frac{\varphi^n}{\sqrt{5}}$? If it is not too far, the we can make our claim stronger by finding the error term. In particular, if $|F_n - \frac{\varphi^n}{\sqrt{5}}| < 1$, since we know that F_n is an integer, we could simply calculate $\frac{\varphi^n}{\sqrt{5}}$ and round it to the nearest integer to get a better and simpler exact formula. Let's calculate:

$$|F_n - \frac{\varphi^n}{\sqrt{5}}| = |\frac{(-\varphi)^{-n}}{\sqrt{5}}|$$

and

$$|\frac{(-\varphi)^{-n}}{\sqrt{5}}| < 1 \iff |(-\varphi)^{-n}| < \sqrt{5}$$

which clearly holds since $\varphi^{-1} < 1$.

Thus, we can safely conclude that

$$F_n = \begin{cases} \lfloor \frac{\varphi^n}{\sqrt{5}} \rfloor, & \text{if } n \text{ is even} \\ \lceil \frac{\varphi^n}{\sqrt{5}} \rceil, & \text{if } n \text{ is odd} \end{cases}$$

which in any case corresponds to rounding $\frac{\varphi^n}{\sqrt{5}}$ to the nearest integer. This is a simpler, more elegant, and computationally cheaper exact formula to calculate F_n.

4 Numerical aspects

We now turn our atention to the actual computation of F_n via Binet's formula, or more precisely, via our improvement of Binet's formula. The reason why this question is worth addressing is because φ, being an irrational number, makes the computation of its powers challenging in terms of numerical accuracy, especially as n grows.

In order to have an good approximation of φ we only need a good approximation of $\sqrt{5}$, which can be obtained as $\lim_{n\to\infty} \frac{a_n}{b_n}$ [1] where

$$a_{i+1} = a_i + 5b_i \tag{3}$$
$$b_{i+1} = a_i + b_i \tag{4}$$
$$a_0 = 2, \;\; b_0 = 1 \tag{5}$$

Now we need to make the computation of φ^n as efficient as possible. One good idea is to use binary exponentiation, which reduces the number of computations of φ^n from $O(n)$ to $O(log_2(n))$ by exploiting the observation that

$$x^n = \begin{cases} (x^2)^{\frac{n}{2}} & \text{if } n \equiv 0 \pmod 2 \\ x(x^2)^{\frac{n-1}{2}} & \text{if } n \equiv 1 \pmod 2 \end{cases} \tag{6}$$

Below is a pseudocode for binary exponentiation.

[1]Further details can be found in my article titled "Approximating square roots using Pascal's triangle", available in my website andrescasillas99.github.io

Algorithm 1 BinExp(x,n)

```
Require: $n \in \mathbb{N}, x \in \mathbb{R}$
  while $n \geq 0$ do
    if $n = 0$ then
      return 1
    else if $n$ is even then
      return BinExp($x^2$, $\frac{n}{2}$)
    else if $n$ is odd then
      return $x*$ BinExp($x^2$, $\frac{n-1}{2}$)
    end if
  end while
```

Not only have we drastically reduced the number of computation needed to calculate powers of φ, but we can further simplify our computations by making use of the mathematical properties of φ, namely that it satisfies the eigenvalue equation 2 ($\varphi^2 = \varphi + 1$). Since this is a Fibonacci-type recurrence, it is not hard to prove (try it!) that

$$\varphi^k = F_{k-1}\varphi + F_{k-2} \tag{7}$$

for all $k > 2$, where F_k is the k-th Fibonacci number.

Moreover, since the powers of φ in binary exponentiation are obtained by iteratively squaring the last term, using the eigenvalue equation 2 one can derive the following property

$$\varphi^{2^k} = A_k\varphi + B_k \tag{8}$$

$$A_{k+1} = A_k^2 + 2A_kB_k \tag{9}$$

$$B_{k+1} = A_k^2 + B_k^2 \tag{10}$$

with initial condition

$$A_1 = 1$$
$$B_1 = 1$$

This comes in handy for calculating the necessary terms in the binary exponentiation algorithm. Notice that from 7 and 8 one can conclude that

$$F_{2^k-1} = A_k$$
$$F_{2^k-2} = B_k$$

where

$$\begin{aligned}
A_{k+1} &= A_k^2 + 2A_k B_k \\
B_{k+1} &= A_k^2 + B_k^2 \\
A_1 &= 1 \\
B_1 &= 1
\end{aligned}$$

Finally, by combining binary exponentiation and equation 7 -when F_{k-1} is small enough to be computed recursively- we can efficiently compute F_n for large values of n.

4.1 Theoretical example

Let us test the power of our method by outlying the computing process of a large Fibonacci number, say 123456. We could simply code a recursive or iterative binary exponentiation method -followed by a simplification of powers of φ by means of equation 7- to get the desired result, but here we outline the "pen-and-paper" steps for a clear outline of the ideas involved.

1. Get the binary expansion of our number, namely $123456_{10} = 11110001001000000_2 = 2^{17} + 2^{16} + 2^{15} + 2^{14} + 2^{10} + 2^7$. This means that
$$\varphi^{123456} = \prod_{k \in \{7,\,10,\,14,\,15,\,16,\,17\}} \varphi^{2^k},$$

2. Compute φ^{2^k} for $k \in \{7,\ 10,\ 14,\ 15,\ 16,\ 17\}$ by iterative squaring i.e.
$$\begin{aligned}
\varphi^{2^2} &= (\varphi^2)^2 \\
\varphi^{2^3} &= (\varphi^{2^2})^2 \\
&\vdots \\
\varphi^{2^{17}} &= (\varphi^{2^{16}})^2
\end{aligned}$$
We could calculate them using recursion 8. If one wants to do it with binary exponentiation, in order to reduce numerical errors, one can use equation 1 on a mildly low power of 2 -say 2^4-, calculate $\varphi^{2^4} = \varphi^{16} = F_{15}\varphi + F_{14} = 987\varphi + 610$, and start squaring from there to obtain the rest of the powers needed.

3. Multiply the relevant terms, in this case $\prod_{k \in \{7,\,10,\,14,\,15,\,16,\,17\}} \varphi^{2^k}$ to get the desired value. Then divide by $\sqrt{5}$ and round to nearest integer. One can use the recursive relationships in 3 to approximate both φ and $\sqrt{5}$.

4.2 Numerical example

Following the previous explanation, let us numerically compute F_{100}.

1. Given that $100 = 2^6 + 2^5 + 2^2$, we proceed to compute compute all relevant powers, namely
   (a) $\varphi^{2^2} = \varphi^4 = F_3\varphi + F_2 = 3\varphi + 2$
   (b) $\varphi^{2^5} = (((3\varphi + 2)^2)^2)^2 = A_5\varphi + B_5 = 2{,}179{,}109\varphi + 1{,}346{,}269$
   (c) $\varphi^{2^6} = (\varphi^{2^5})^2 = 10{,}618{,}467{,}002{,}743\,\varphi + 6{,}563{,}949{,}423{,}242$
2. Using equation 3 to approximate $\sqrt{5}$ (with 15 terms), we obtain $\sqrt{5} \approx \frac{2{,}889}{1{,}292}$ in which case $\varphi \approx \frac{1568}{969}$. Our desired result is then

$$
\begin{aligned}
F_{100} &= \lfloor \frac{\varphi^{100}}{\sqrt{5}} \rfloor \\
&\approx \lfloor \frac{1{,}292}{2{,}889}(3\frac{1568}{969} + 2)(2{,}179{,}109\frac{1568}{969} + 1{,}346{,}269)(10{,}618{,}467{,}002{,}743\,\frac{1568}{969} + 6{,}563{,}949{,}423{,}242)\rfloor \\
&= 354, 541, 420, 533, 205, 195, 436
\end{aligned}
$$

which is about 0.09% larger than $F_{100} = 354, 224, 848, 179, 261, 915, 075$.

5 Conclusion

The previous algorithm might seem more complicated and inaccurate for moderate values of n (such as $n = 100$), in which case it is indeed better and faster to use the classical Fibonacci recursion 1. However, for very large values of n (say, for example, $n > 10^7$) Binet's formula and binary exponentiation really pay off. One further remark to be made is that we have calculated absolutely everything, even approximations to $\sqrt{5}$ and φ, which add to our error term in the final computation. These approximations, however, are no challenge in modern numerical mathematics and could be assumed to be as precise as needed.