MATH 100 – Introduction to the Profession

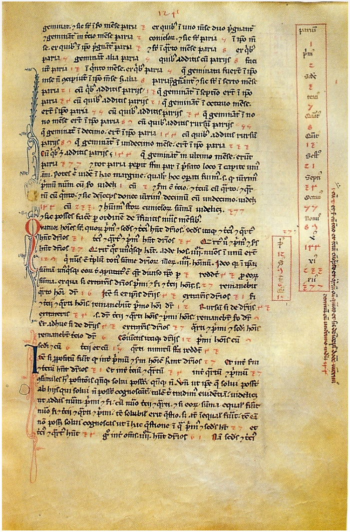
Vectors, Functions and Dates in MATLAB (Fibonacci Numbers and Calendars)

Greg Fasshauer

Department of Applied Mathematics
Illinois Institute of Technology

Fall 2012

Fibonacci[1] Numbers



[1] Fibonacci brought Arabic numerals into Western culture.

- Start with two newborn rabbits (one male rabbit and one female)[2].
- A rabbit will reach sexual maturity after one month.
- The gestation period of a rabbit is one month.
- Once it has reached sexual maturity, a female rabbit will give birth to exactly one male and one female rabbit every month.
- Rabbits never die.

How many pairs will there be at the end of one year?

Leonardo Fibonacci, *Liber Abaci* (1202)

Run the Mathematica demo `FibonacciRabbits.cdf`.

Remark

Even though this problem has been around since 1202, it's just a "textbook problem". Rabbits do die, and they don't reach maturity in one month (it's more like 6 months), etc..

[2]Note that [ExM] starts with a **mature** pair of rabbits, i.e., the sequence there begins with $f_1 = 1$, $f_2 = 2$.

To have some reasonable notation, we let f_n denote the number of rabbit pairs at the beginning of the n^{th} month.

Since it takes one month for a newly born pair to mature, the sequence begins with

$$f_1 = 1, \qquad f_2 = 1.$$

After that, the sequence progresses as

$$f_n = f_{n-1} + f_{n-2},$$

i.e., the number of rabbits in a new month, f_n, consists of those who were alive a month ago, f_{n-1}, and the babies of those who were also around 2 months ago (i.e., were mature), f_{n-2}.

As mentioned earlier, if we don't want to enter all commands interactively in the MATLAB command window, then we can use M-files.

An M-file can be a script (such as `scavenger_assign.m`) or `fibonacci13.m`:

```
% FIBONACCI13
% Generates the first 13 Fibonacci numbers
f = [1 1]
for n=3:13
    f(n) = f(n-1) + f(n-2)
end
```

Remark

- *Here we use a* `for-loop` *to iteratively compute the first 13 Fibonacci numbers and store them in the vector* `f`*.*
- *Note that* `f` *is expanded as needed. This can be inefficient, but eliminates the need to allocate memory.*

Or an M-file can be a function such as `fibonacci.m`:

```
function f = fibonacci(n)
% f = FIBONACCI(n)
% Generates the first n Fibonacci numbers
f = zeros(n,1)
f(1) = 1
f(2) = 1
for k = 3:n
    f(k) = f(k-1) + f(k-2)
end
```

Remark

- *This function is similar to the previous script. However, it allows us to specify an upper limit for the* `for-loop` *without having to rewrite the code.*
- *Here we did allocate memory for* `f`*.*

Now a recursive function:

```
function f = fibnum(n)
%FIBNUM  Fibonacci number.
%   FIBNUM(n) demonstrates recursion by generating the
%   Warning: FIBNUM(50) takes a very long time.
if n <= 2
   f = 1;
else
   f = fibnum(n-1) + fibnum(n-2);
end
```

Remark

- *Note that we use an* `if...else` *conditional to handle the end of the recursion.*
- *Also note that the function calls itself with smaller values of n (⇝ recursion).*

Example

Recursion is generally[a] slower than iteration:

```
tic, fibonacci(20), toc
tic, fibnum(20), toc
```

[a] This depends on the programming language.

Remark

Recursion is essential in the design of so-called divide-and-conquer algorithms.

Fibonacci Numbers and the Golden Ratio

Run the Mathematica demo
`FibonacciNumbersAndTheGoldenRatio.cdf`.

We can solve the difference equation (or recursion)

$$f_n = f_{n-1} + f_{n-2} \tag{1}$$

by using the *Ansatz*

$$f_n = cx^n \tag{2}$$

for some yet to be determined numbers x and c. Then $f_{n-1} = cx^{n-1}$ and $f_{n-2} = cx^{n-2}$ so that we get

$$f_n = f_{n-1} + f_{n-2} \iff cx^n = cx^{n-1} + cx^{n-2} \iff cx^{n-2}x^2 = cx^{n-2}x + cx^{n-2}$$

or (assuming $cx^{n-2} \neq 0$)

$$x^2 = x + 1.$$

Recall that the solutions of the quadratic equation

$$x^2 = x + 1$$

are $x_1 = \phi$ (the golden ratio) and $x_2 = 1 - \phi$.

Plugging the two solutions x_1 and x_2 into our *Ansatz* (2), we get two possible solutions $x_1^n = \phi^n$ and $x_2^n = (1-\phi)^n$, which can be used to obtain all possible solutions of (1) via linear combinations, i.e.,

$$f_n = c_1 \phi^n + c_2 (1-\phi)^n. \tag{3}$$

Since, however, we want a very special solution (namely the one for which $f_1 = f_2 = 1$), we end up with two conditions that will determine the constants c_1 and c_2:

$$f_1 = c_1\phi + c_2(1-\phi) \stackrel{!}{=} 1$$

$$f_2 = c_1\phi^2 + c_2(1-\phi)^2 \stackrel{!}{=} 1.$$

You will use MATLAB to solve these equations in HW 6 (Exercise 2.3), but one can also find the constants by hand as $c_1 = \frac{1}{2\phi-1}$ and $c_2 = \frac{1}{1-2\phi}$, and then get the solution of (1) from (3).

The following MATLAB code computes the first 12 Fibonacci numbers by directly evaluating the formula just derived :

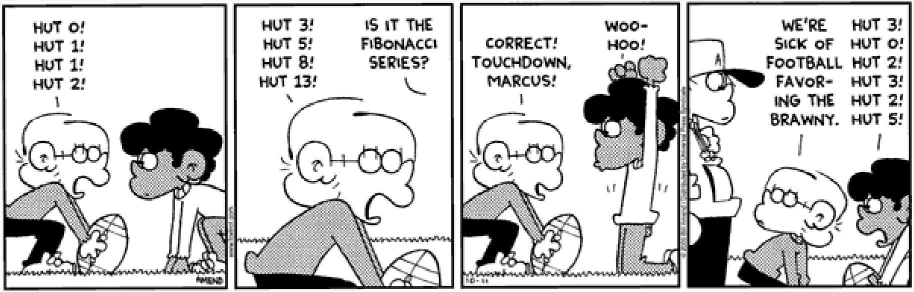
```
n = (1:13)';
phi = (1+sqrt(5))/2
f = (phi.^n - (1-phi).^n)/(2*phi-1)
```

Remark

- *Note the elementwise operator* `.^` *is used to compute the power of ϕ^n for all different values of n simultaneously.*
- *To get "clean" integer values we could use* `round(f)`, `floor(f)` *or* `fix(f)`.

Applications

Applications



Look through `fibonacci_recap.m`.

Friday the 13th

Read the corresponding section in [ExM] and look at `friday13.m`.

References I

- T. A. Driscoll.
  Learning MATLAB.
  SIAM, Philadelphia, 2009.
  http://epubs.siam.org/ebooks/siam/other_titles_in_applied_mathematics/ot115
- D. J. Higham and N. J. Higham.
  MATLAB Guide.
  SIAM (2nd ed.), Philadelphia, 2005.
  http://epubs.siam.org/ebooks/siam/other_titles_in_applied_mathematics/ot92
- C. Moler.
  Numerical Computing with MATLAB.
  SIAM, Philadelphia, 2004.
  http://www.mathworks.com/moler/index_ncm.html

References II

C. Moler.
Experiments with MATLAB.
Free download at
http://www.mathworks.com/moler/exm/chapters.html

L. Sigler.
Fibonaci's Liber Abaci.
Springer (New York), 2003.

The MathWorks.
MATLAB 7: Getting Started Guide.
http://www.mathworks.com/access/helpdesk/help/pdf_doc/matlab/getstart.pdf