FROM DIV GRAD CURL TO FIBONACCI

OLIVER KNILL

Abstract. How many meaningful expressions in div, curl and grad are there?

1. Div, Curl and Grad

1.1. The **divergence**, the gradient and the curl are the three fundamental derivative operations in multi-variable calculus of three dimensional space. [3]. They incarnate the exterior derivative in dimension 3. In multivariable calculus, where zero and three forms are identified and one and two forms are identified, the gradient maps scalar fields to vector fields, the curl maps vector fields to vector fields and the div maps vector fields to scalar fields.

1.2. Not all combinations make sense. One can not form **grad grad** nor **div div** for example but **curl curl** is possible. From the nine possible expressions, the five **div curl**, **div grad**, **curl curl**, **curl grad**, and **grad div** make sense and the four **grad grad**, **curl div**, **grad curl div div** are not defined in this frame work.

1.3. In a recent multivariable calculus final exam we asked students to identify which of the 27 configurations of length 3 make sense. As many figured out, there are now 8 cases which makes sense and 19 which do not. You can do that exercise too. It is a bit of a grind, but when doing that rather tedious task, one can witness some patters like that that one can cross out any occurance of **div div** for example.

1.4. One of the reasons to pose this problem was because it leads to the combinatorial problem of finding the number of meaningful combinations of length k. From the 81 cases for $k = 4$, there are 13 which make sense. Now, knowing that in the case $k = 1$, all three expressions make sense, we have already the sequence $f(1) = 3, f(2) = 5, f(3) = 8, f(4) = 13, ...$ How does it continue?

2. Fibonacci

2.1. As the cryptologist Sophie Neveu from the novel "the Da Vinci Code" [1] would immediately suspect that this is part of the Fibonacci sequence. Indeed:

Theorem 1. *The number $f(k)$ of meaningful expressions of length k one can form with div, curl and grad is the Fibonacci number $F(3 + k)$.*

Date: 12/14/2019.
1991 *Mathematics Subject Classification.*

Proof. Let $a(k)$ be the number of words of length k which start with div and $b(k)$ the number of words of length k which start with curl and $c(k)$ the number of words which start with grad. We have $c(k) = a(k-1)$, $b(k) = c(k-1) + b(k-1)$ and $a(k) = c(k-1)$+b(k-1). This leads to the discrete recursion

$$v(k) = \begin{bmatrix} a(k) \\ b(k) \\ c(k) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} a(k-1) \\ b(k-1) \\ c(k-1) \end{bmatrix} = Av(k-1)$$

The eigenvalues of A are the golden mean $\phi = (1+\sqrt{5})/2, \phi^{-1}, 0$. Writing down the closed form solution of the discrete dynamical system starting with $v(1) = [1,1,1]$ gives a solution $v(k)$ which satisfies $a(k) + b(k) + c(k) = F(k+3)$ as one can verify with the Binet formula. □

2.2. One can compute the matrix entries of the k'th iterate directly and see the Fibonacci numbers

$$A^k = \begin{bmatrix} F(k-1) & F(k) & F(k) \\ F(k-1) & F(k) & F(k) \\ F(k-2)) & F(k-1) & F(k-1) \end{bmatrix}$$

Which can be proven by induction using the recursion $A^k = AA^{k-1}$ and that it works for $k = 2$. This holds also for $k = 1$ if one defines $F(-1) = 1$. The total number of parts is now $F(k+3) = F(k+2) + F(k+1) = 2F(k+1) + F(k)$.

3. A digraph

3.1. A directed graph is a finite set of nodes with finitely many connections, in which self loops and multiple connections are allowed. The matrix A can be interpreted as the adjacency matrix of a digraph. This **div-curl-grad** digraph is shown in the following picture:

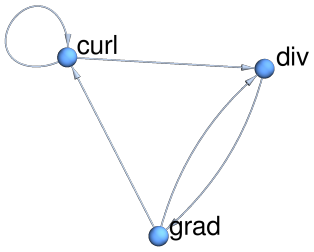

FIGURE 1. The div-grad-curl graph encodes which transitions are possible. We can form *curlgrad* and *divgrad*, *graddiv* and *divcurl* as well as *curlcurl*. The graph has 5 directed edges therefore. The adjacency matrix of this graph has the golden ratio as an eigenvalue.

3.2. We can now say that:

Corollary 1. *There are $F(k+3)$ paths of length k in the div-curl-grad digraph.*

3.3. We can also associate to this graph a Markov process. Assume we are do a random walk, jumping from g to d and jumping from each of the nodes d and c with probability to either g or c. The corresponding doubly stochastic matrix is

$$P = \begin{bmatrix} 0 & 1/2 & 1/2 \\ 0 & 1/2 & 1/2 \\ 1 & 0 & 0 \end{bmatrix} .$$

Its Perron Frobenius eigenvector is defined as the eigenvector $[1,1,1]$ to the eigenvalue 1. It is unique in this case. We see that a random walker in that graph spends about equal time in on each node in average.

3.4. If we look at the space X of all possible sequences in the div-curl-grad language, we get compact topological space on which the shift map $T(x)_k = x_{k+1}$ acts. It carries a natural invariant measure μ. One calls such a system a **Markov chain**. Its entropy is $\log(\lambda)$, where λ is the maximal eigenvalue.

Corollary 2. *The entropy of the div-curl-grad Markov chain is* $\log(\phi)$.

One tells that the system is chaotic. There are other spaces X on which the shift is not chaotic. Some of them are given by grammars too. One is interested in such sequences for example in solid state physics, where one can use the values $x(k)$ as potentials for a Schrödinger operotor.

4. Grammar

4.1. A **grammar** in a language is a set of rules. One can then look at the number of possible words of length k in this language. In our case, the alphabet consists of three letters $A = \{d, c, g\}$ which stands for div, curl and grad. The grammar allows for the transitions $G = \{dc, cg, dg, gd\}$. We have answered:

Corollary 3. *There are $F(k+3)$ words in the div-curl-grad language.*

4.2. Our spoken language is described by complex grammars. Our genetic code is described in a genomic language which is known to be even more complex. When studying ancient languages like the Khipu language, one has to look at sequences of knot patters to reconstruct the meaning. Of course, our little riddle with div-curl-grad is much simpler but the task is similar. One first produces a database of texts (this is what our students had to do in the exam) and then do a lot of counting.

4.3. The basic statistical cryptological attack to a text is to count letters and then compare the frequencies. This allows to break simple substitution cyphers like the Caesar code, Rot 13 or Altbash.

Corollary 4. *In the div-curl-grad language, all letters appear with the same frequency in average*

5. Probabilities

5.1. When writing down a differential operator expression involving div, curl and grad of length k, the probability to get something meaningful is $p(k) = F(k+3)/3^k$.

Corollary 5. *A monkey typing randomly k letters hits with probability $p(k) = F(k+3)/3^k$ a word in the div-curl-grad language.*

5.2. Already when writing words of length 10 the chance is about 0.4 percent to hit a meaningful expression and words of length 100 give something meaningful only in one of $3 \cdot 10^{-27}$ cases. This is already extremely small. This is the chance to hit a specific water molecule in 3 gallons of water.

5.3. Problems like this are also interesting on a meta level. How does one come up with interesting problems and how does one suspect a result at first? History shows that break through ideas often come from rather long grinding periods, where one studies cases. Our students have had to grind through many cases. Math educators in general abhor such exercises and also in our group, we had arguments whether this problem was a good one. Maybe to provoke a bit the education community (to which I belong too) one can argue that "grinding through cases" is an integral part of science. 99 percent of any research is repetitive work and break through moments come often only after long searches [2]. Repetitive work also has a meditative aspect. Much of the frustration which educators have imposed on our students (especially the young ones) is to avoid repetition and to expect that every computation is done in some clever way.

5.4. Along that line, I did get the Fibonacci connection only by detour and a bit of luck: I originally was not looking at the good cases but directly wondered about the probability that an expression makes sense and to investigate how the probability $p(k)$ goes to zero, I data fitted $-\log(p(k)) \sim b+mk$ (of course after writing a program which computes all possible cases of length k) and ended up with a number 0.6174 which appeared to be close to the golden ratio ϕ.

5.5. Here is the program I used (before seeing any connection with Fibonacci). The nice thing about high level programming languages is that this can be done in 15 minutes, while writing a Python or C program would need hours:

```
e={{d,c},{d,g},{g,d},{c,g},{c,c}};
f[x_]:=Module[{t=True},Do[t=t&& MemberQ[e,{x[[k]],x[[k+1]]}],{k,Length[x]-1}];t];
G[n_]:=Module[{},X=Map[f,Tuples[{d,c,g},n]];Length[Position[X,True]]/3^n];
Fit[-Log[Table[G[n],{n,2,10}]], {1, x}, x]
```

5.6. Of course, after seeing the pattern and writing down the graph and adjacency matrix one can achieve the same much simpler. Here is a linear fit for words of length 100 (but the following program was written down after the analysis was down):

```
A={{0,1,1},{0,1,1},{1,0,0}};
G[k_]:=Last[Map[Total,MatrixPower[A,k+2]]]/3^k
Fit[-Log[Table[G[n],{n,2,100}]], {1, x}, x]
```

The ultimate computation is

```
G[k_]:=Fibonacci[k+3]/3^k
```

5.7. Back to the data fit, I got disappointed to see that the best fit $-0.02373 + 0.617681$ does not quite have the golden ratio slope even when computing for larger k like 14, where almost 5 million sentences had to be checked. Indeed, the slope is $\log(\phi) - \log(3) = -0.6174...$, computed as $N[Log[GoldenRatio] - Log[3]]$. Because even computer algebra systems had difficulty with larger k, I started to look at the actual integer list $G[n]$ of cases which makes sense to see a pattern. Unlike in the novel of Dan Brown, looking at three numbers like $3, 5, 8, 13$ did not immediately ring a bell and associate Fibonacci to me, especially because the 1,1,2 part are missing. So, because $\log(3)$ is close to 1, I was lucky to get the suspicion about ϕ using linear regression. In any case, this little story illustrates that discoveries are mostly luck but (and this makes the point of Perkins) only come after a longer period of search.

References

[1] D. Brown. *The Da Vinci Code*. Random House, 2000.
[2] D. Perkins. *The Eureka Effect*. WW. Nortaon, New York, London, 2000.
[3] H.M. Schey. *div grad curl and all that*. Norton and Company, New York, 1973-1997.

Department of Mathematics, Harvard University, Cambridge, MA, 02138