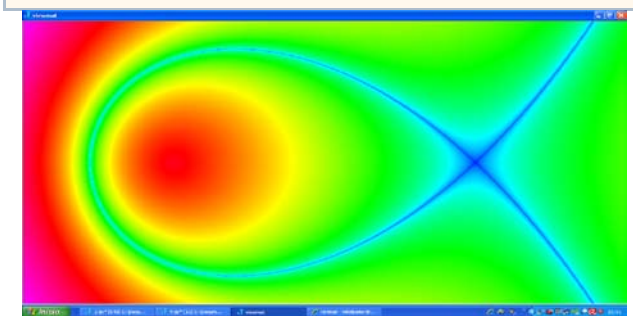
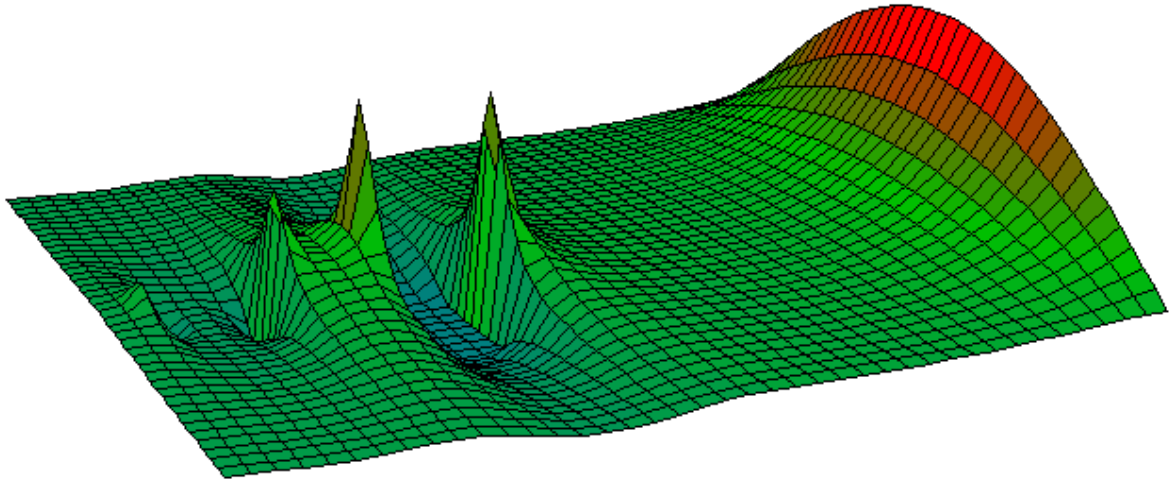
Journal of J

An interdisciplinary journal on J programming language and applications in science and technology.





Please Circulate

Journal of J.

MPM press

An open access Journal

ISSN: 2174-9280

Vol.1, No.3 October 2012

JoJ (J2-team)

J, a language for the science, technology and more.

Aims and scope

Journal of J is a **non professional and nonprofit** journal , involving a large research and users community in J programming language.

Journal of J is an interdisciplinary **journal** devoted to J and science and technology.

Journal of J aims to provide fast access to papers about science, technology and J.

Journal of J embodies the following principle:

Open Access: Knowledge is a public good. All readers have open access to reading and downloading papers. The simple and free access ensures maximum readership and high citation records for published papers.

Contact: journalofj at hotmail dot com

Style and Contents

Journal of J aims to cover all the main areas of science. Inevitably, articles in different areas are addressed at different audiences. Many of the articles submitted to the journal are standard technical pieces, addressed to a purely academic audience

To attract this variety of contributions *Journal of J* will contain the following areas:

- Mathematics: number theory, logic, calculus, algebra, arithmetic, algorithms and others…
- Physics: dynamical systems, chaos, fractals, disorder, statistical physics and others…
- Computer science, Visualization, Engineering, Computer Art, …

Visit:

http://sites.google.com/site/jforscience/

Edited by Mikel Paternain

Editorial

En este número contamos nuevamente con unas interesantes contribuciones de fuerte componente matemático.

Como dialecto proveniente de APL, J es un lenguaje de marcada base matemática y especialmente útil para sintetizar ideas en esta área.

Contamos en este número con unas interesantes contribuciones de Roger Hui, colaborador de K. Iverson y co-desarrollador del lenguaje J. En este trabajo el profesor Hui nos habla del primer artículo publicado por K. Iverson lo que es una interesantísima contribución a la bibliografía. El profesor Cliff Reiter nos presenta de forma de artículo las anotaciones que empleó en la Conferencia sobre el lenguaje J desarrollada en Toronto el pasado mes de Julio.

Bo Jacoby presenta algunas cuestiones interesantes sobre deducción, inducción, etc. El alcance y validez de sus afirmaciones está por determinar.

Dado que la mayoría de las contribuciones en el lenguaje se distribuyen y dan a conocer en el foro de J (origen de JoJ) pocas contribuciones llegan a nuestra redacción todavía. No obstante esperamos que el nivel de las contribuciones siga aumentando.

Mikel Paternain (Editor)

mikelpater at Hotmail dot es

Short NOTE:

The Zeros of the Partial Sums of e^z

Roger Hui

Reference 26 of Ken Iverson's Ph.D. thesis [0] was:

> Iverson, K.E., "The Zeros of the Partial Sums of e^z ", *Mathematical Tables and Other Aids to Computation* 7 (1953).

I quickly found a citation [1] on the internet:

```
@Article{Iverson:1953:ZPS,
  author =    "K. E. Iverson",
  title =     "The Zeros of the Partial Sums of $e^z$",
  journal =   j-MATH-TABLES-OTHER-AIDS-COMPUT,
  volume =    "7",
  number =    "43",
  pages =     "165--168",
  month =     jul,
  year =      "1953",
  CODEN =     "MTTCAS",
  ISSN =      "0891-6837",
```

```
  bibdate =   "Tue Oct 13 08:06:19 MDT 1998",
  bibsource = "http://www.math.utah.edu/pub/tex/bib/
                  mathcomp1950.bib; JSTOR database",
  acknowledgement = ack-nhfb,
  fjournal =  "Mathematical Tables and Other Aids
                  to Computation",}
```

From thence a helpful J Forum member provided a link to the actual text [2]. This is the earliest Ken Iverson publication I have found so far. The paper began thus:

The location of the zeros of certain entire functions has received considerable attention in the literature, and it was suggested by R.S. Varga that a table of the zeros of certain truncated power series might be of interest. Accordingly, the zeros of the truncated exponential series $S_n(z)$ = *****$z^k/k!$ have been computed for values of n up to 23. They appear in the accompanying table with the complex zeros ordered as to modulus and with the

The table entry for n = 23 stated:

– 7.1926 8590 7451 ± 1.4750 5919 5082 *i*
– 6.8443 1411 8665 ± 2.9355 2075 2798 *i*
– 6.2551 3130 4907 ± 4.3657 9349 6476 *i*
– 5.4112 0644 7035 ± 5.7481 1045 3530 *i*
– 4.2905 2911 0822 ± 7.0608 9517 9930 *i*
– 2.8595 2302 8914 ± 8.2762 1397 0162 *i*
– 1.0664 4595 7935 ± 9.3553 6015 8909 *i*
1.1720 9815 7227 ± 10.2403 1777 0398 *i*
4.0025 2534 8326 ± 10.8348 6485 1671 *i*
7.7243 3865 4741 ± 10.9536 0414 4523 *i*
13.1748 6483 8907 ± 10.1262 6417 8727 *i*
– 7.3079 8221 4646

The verb `p.` in J [3] works with polynomials. The dyad `x p. y` evaluates the polynomial `x` at `y`; the monad `p. x` converts `x` between coefficients `c` and multiplier and roots `(m;r)`. The coefficients of the partial sum of e^z for n=23 is:

```
   c=: % ! i.24
   c
1 1 0.5 0.166667 0.0416667 0.00833333 0.00138889 ...
```

Converting from coefficients to muliplier and roots:

```
   mr=: p. c
   mr
┌──────────┬──────────────────────────────────────────────────┐
│3.86817e_23│13.1749j10.1263 13.1749j_10.1263 7.72434j10.9536 ...│
└──────────┴──────────────────────────────────────────────────┘
   r=: > {: mr
   r
13.1749j10.1263 13.1749j_10.1263 7.72434j10.9536 ...
```

The roots from 1953 are ordered in increasing modulus, with the real root last; the roots computed by J are ordered in decreasing modulus. We compare the evaluation of the roots from 1953 and of the reordered roots computed by J:

```
   r1953=: _7.192685907451j1.475059195082 ...
   _5 ]\ | c p. r1953
7.36698e_14 7.36698e_14 1.37043e_13 1.37043e_13 1.20857e_13
1.20857e_13 6.67108e_14 6.67108e_14 8.19945e_14 8.19945e_14
1.53714e_13 1.53714e_13 2.75251e_13 2.75251e_13 6.80251e_12
6.80251e_12 1.12317e_10 1.12317e_10   3.0922e_9   3.0922e_9
 5.55512e_8  5.55512e_8 9.19177e_11           0           0

   _5 ]\ | c p. 1 |. r /: | r
3.76597e_14 3.76597e_14  4.7101e_14  4.7101e_14  2.6361e_14
 2.6361e_14 5.01272e_14 5.01272e_14 9.52945e_14 9.52945e_14
 3.9363e_14  3.9363e_14 2.26531e_13 2.26531e_13 3.41104e_13
3.41104e_13 3.58005e_12 3.58005e_12 3.33604e_11 3.33604e_11
 3.7143e_10  3.7143e_10 1.73195e_14           0           0
```

I survey the last results with considerable relief. It would have been shameful to produce a worse result 59 years later.

References

- [0] Iverson, K.E., *Machine Solutions of Linear Differential Equations — Applications to a Dynamic Economic Model*, Doctoral Thesis, Harvard University, 1954-01. http://www.jsoftware.com/papers/MSLDE.htm
- [1] http://ftp.math.utah.edu/pub//tex/bib/mathcomp1950.html#Iverson:1953:ZPS
- [2] http://www.ams.org/journals/mcom/1953-07-043/ S0025-5718-1953-0057013-0/S0025-5718-1953-0057013-0.pdf
- [3] Hui, R.K.W., and Iverson, K.E., *J Introduction and Dictionary*, Jsoftware Inc., 2012. http://www.jsoftware.com/help/dictionary/dpdot.htm

Paper:

The Story of Fractals, Visualization and J

Cliff Reiter

Lafayette College, Department of Mathematics, Easton PA, 18042 U.S.A.

These notes roughly follow the talk given at the 2012 Jsoftware conference in Toronto, July 24, 2012. They are based upon 32 bit J6.02 under Windows on a PC with the *media\image3* and graphics\fvj3 addons installed.

I started teaching mathematics at Lafayette College in 1983 and in the late 1980s the college did not have a computer science major. I offered a Turbo Pascal based special topics graphics course and soon found myself asking student's to do many projects involving graphics fractals. Some of these led to my earliest collaborations with students as in the links below and Figure 1.

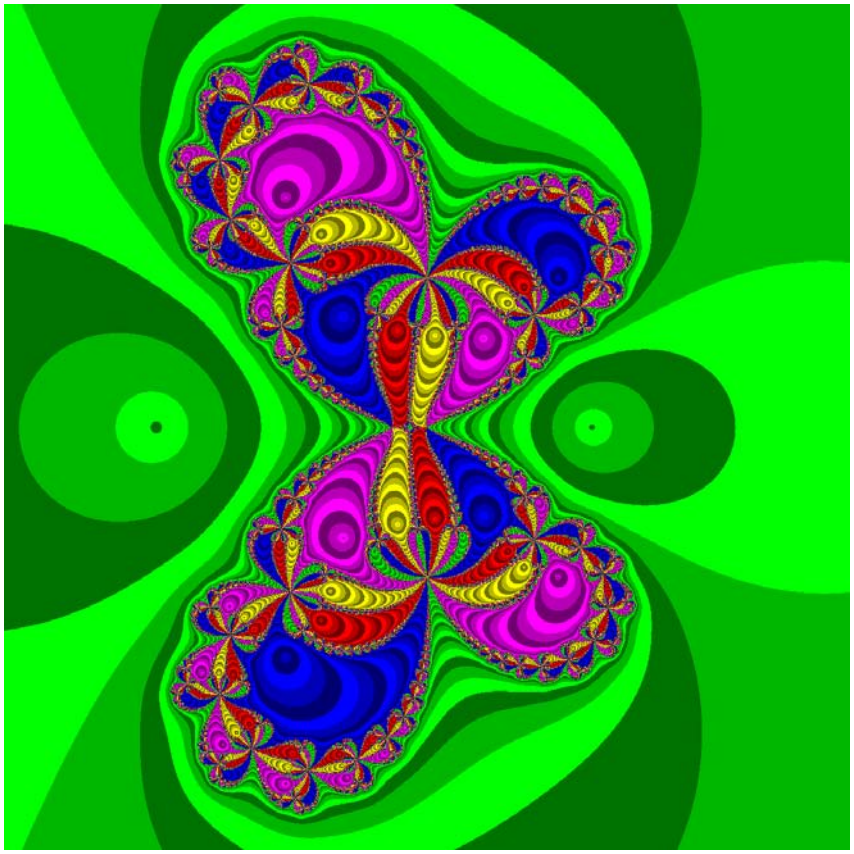


Figure 1. A butterfly from Newton's Method on Systems

http://webbox.lafayette.edu/~reiterc/mvp/newt_sys/index.html

http://webbox.lafayette.edu/~reiterc/mvp/rqi/index.html

http://webbox.lafayette.edu/~reiterc/mvp/ec_julia/index.html

In the same era I also taught a second linear Linear Algebra class using APL and one of the students remarked that APL would be really good for computer graphics. Indeed, some fractal constructions are easy in APL and J but difficult in Pascal.

For example, juxtaposition fractals would be awkward to create in Pascal but are easy in APL or J. Below we see the fractal built by placing an array beside itself and above itself with the corner padded. A Sierpinski triangle is formed as seen in Figure 2.

```
load '~addons\media\image3\view_m.ijs'

load '~addons\media\image3\prevare.ijs'

spix=:[ # #"_1

    (,,.~)i.2 2

0 1 0 0

2 3 0 0

0 1 0 1

2 3 2 3

   (,,.~)^:4,1

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0

1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0

1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0

1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0

1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0

1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0

1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0

1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0

1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0

1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
```

```
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

view_data 10 spix (,,.~)^:6,1
```

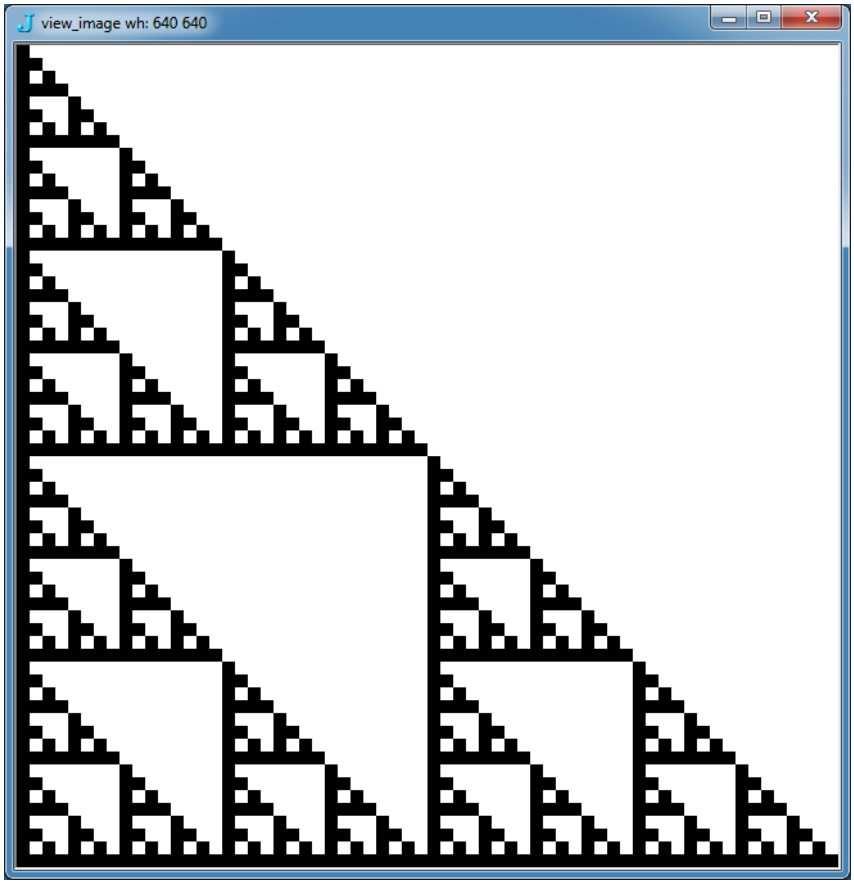


Figure 2. Sierpinski Triangle as a Juxtaposition Fractal

1. FRACTALS FROM INDICES

I soon became interested in fractals created by indices. In particular, gcd's of pairwise lcm's in various bases and in both 2d and 3d seemeded to be very interesting and I wanted ot to be able to write about them. Below we see a table of pairwise lcm's of 3 digit binary addresses. The other experiments are worth experimenting with. One is shown in Figure 3.

```
   ]M=:#:i.8
0 0 0
0 0 1
0 1 0
0 1 1
1 0 0
1 0 1
1 1 0
1 1 1

   M <"1@:|: . *. |:M

+-----+-----+-----+-----+-----+-----+-----+-----+
|0 0 0|0 0 0|0 0 0|0 0 0|0 0 0|0 0 0|0 0 0|0 0 0|
+-----+-----+-----+-----+-----+-----+-----+-----+
|0 0 0|0 0 1|0 0 0|0 0 1|0 0 0|0 0 1|0 0 0|0 0 1|
+-----+-----+-----+-----+-----+-----+-----+-----+
```

```
|0 0 0|0 0 0|0 1 0|0 1 0|0 0 0|0 0 0|0 1 0|0 1 0|
+-----+-----+-----+-----+-----+-----+-----+-----+
|0 0 0|0 0 1|0 1 0|0 1 1|0 0 0|0 0 1|0 1 0|0 1 1|
+-----+-----+-----+-----+-----+-----+-----+-----+
|0 0 0|0 0 0|0 0 0|0 0 0|1 0 0|1 0 0|1 0 0|1 0 0|
+-----+-----+-----+-----+-----+-----+-----+-----+
|0 0 0|0 0 1|0 0 0|0 0 1|1 0 0|1 0 1|1 0 0|1 0 1|
+-----+-----+-----+-----+-----+-----+-----+-----+
|0 0 0|0 0 0|0 1 0|0 1 0|1 0 0|1 0 0|1 1 0|1 1 0|
+-----+-----+-----+-----+-----+-----+-----+-----+
|0 0 0|0 0 1|0 1 0|0 1 1|1 0 0|1 0 1|1 1 0|1 1 1|
+-----+-----+-----+-----+-----+-----+-----+-----+
```

```
   M  +./ . *. |: M
0 0 0 0 0 0 0 0
0 1 0 1 0 1 0 1
0 0 1 1 0 0 1 1
0 1 1 1 0 1 1 1
0 0 0 0 1 1 1 1
0 1 0 1 1 1 1 1
0 0 1 1 1 1 1 1
0 1 1 1 1 1 1 1
```

```
   view_data 5 spix M  +./ . *. |: M=:#:i.2^7

   P256 view_data 5 spix M  +/ . *. |: M=:#:i.2^7

   P256 view_data 5 spix M  +./ . *. |: M=:(5#3)#:i.3^5

   P256 view_data 5 spix M  +/ . *. |: M=:(5#3)#:i.3^5

   P256 view_data 5 spix M  >./ . <. |: M=:(5#3)#:i.3^5

   P256 view_data 5 spix M  i.&1"1@|: . *. |: M=:#:i.2^7

   P256 view_data 5 spix M  i.&1"1@|: . *. |: M=:(5#3)#:i.3^5

   P256 view_data 5 spix M  i.&2"1@|: . *. |: M=:(5#3)#:i.3^5

   P256 view_data 5 spix M  i:&1"1@|: . *. |: M=:(5#3)#:i.3^5
```

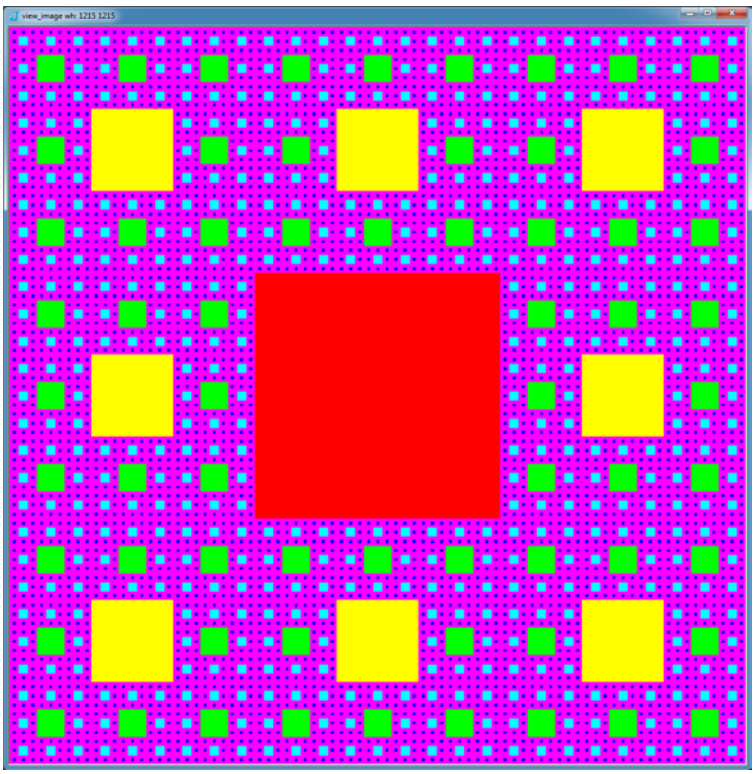


Figure 3. Sierpinski Carpet from indices

However, this was in an era when typing the APL characters was nontrivial. In fact, the system I was using broke under DOS 4.0. It wasn't apparent how to use WordPerfect alternatives to type APL characters. I remembered J was ASCII and, moreover, I vaguely remembered some discussion of inner product being generalized in J. Several of my constructions were anomalous since nested arrays were required. I soon realized that J would not only make the typesetting easy, but it allowed me to simplify the presentation.

That gave the fractals and J. As the computer science program at Lafayette College became robust, I no longer needed to teach computer graphics and could thus emphasis visualizing mathematical ideas. Eventually Ken Iverson agreed to publish a book I could teach the visualization course from resulting in "Fractals, Visualization and J".

2. FRACTALS WITH HURST EXPONETS

Hurst studied the flooding of the Nile and noted that high flows often persisted for years rather than having a Gaussian distribution. Below is a random Gaussian walk and then some with various Hurst exponents. Antipersistent walks occur for Hurst exponent $0<H<0.5$. Gaussian walks occur for $H=0.5$ and persistent walks occur for $0.5<H<1.0$. Examples of slightly antipersistent and persistent walks are shown in Figures 4-5. Other experiments below are worth a try.

```
load 'plot'

   randunif=: ?@($&0) : ({.@[+({:-{.)@[*$:@])

   randsn=: cos@+:@o.@randunif * %:@-@+:@^.@randunif

   setseed=: 9!:1
```

```
   delay=:6!:3

   'pensize 3' plot +/\randsn 300

   interp=: (}. + }:)@:(2:#-:)

   interp 1 2 5
1 1.5 2 3.5 5

   osz=: %:@-.@(2&^)@+:@<:@[

   0.5 osz 1 1 1
0.707107

   0.1 osz 1 1 1
0.84429

   sz=: osz * %@+:@<:@#@] ^ [

   0.5 sz 1 1
0.5

   0.5 sz 1 1 1
0.353553

   0.1 sz 1 1 1
0.734997

   randadd=: ] + sz * randsn@$@]

   hwalk=: 4 : 'x([randadd interp@])^:(y) 0,(x osz 1)*randsn 1'

   'pensize 2' plot 0.1 hwalk 8

   'pensize 2' plot 0.3 hwalk 8

   'pensize 2' plot 0.5 hwalk 8

   'pensize 2' plot 0.7 hwalk 8

   'pensize 2' plot 0.9 hwalk 8

   (3 : 'plot y hwalk 8[setseed 7^5[delay 0.1')"0] 0.01*1+i.99
```

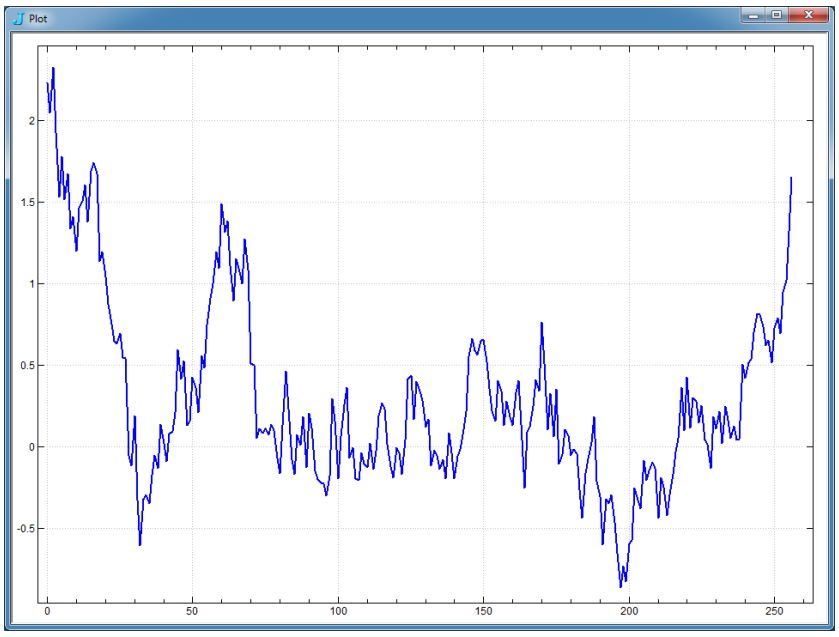


Figure 4. Random Walk with Hurst Exponent 0.3

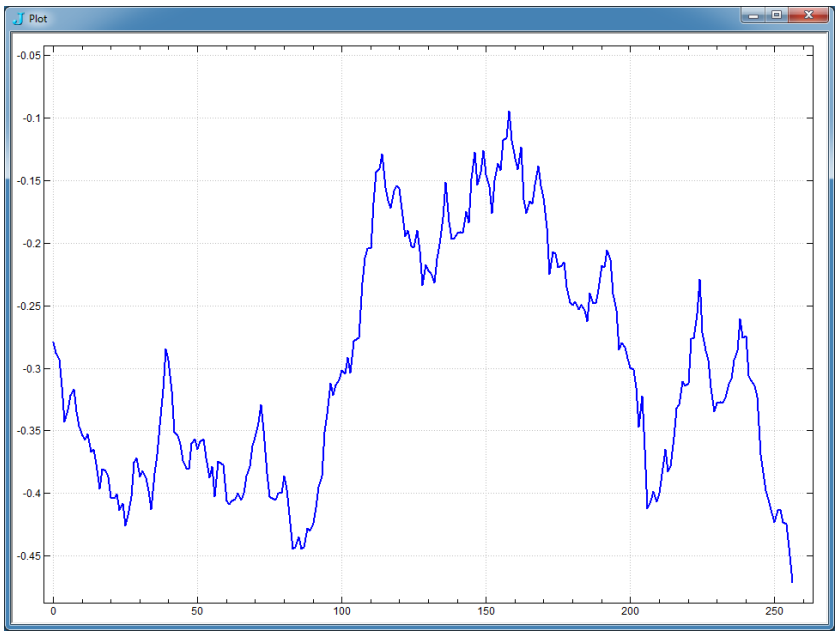


Figure 5. Random Walk with Hurst Exponent 0.7

Of course if we can do a fractal walk in 1D we can do it 2D too. Note the similarity of hwalk with fm. The main difference isn't 1D versus 2D, but rather how initial data is handled. Figure 6 shows a slightly antipersistent fractal mountain.

```
    load '~addons\graphics\fvj3\povkit2.ijs'

   povpath=:'d:\temp\pov\'

   interp2
interp"1@:interp

      fm=: 1 : 0
:
x ([randadd interp2@])^:y m
)
```

```
      m0=: ".;._2]0 : 0
_3  _3    2.2  1    _2
_1   1.5  1    0.5  _1
_0.5 1   _1   _2    _2
_1   0.5 _0.5  0.8  _1
_3  _2.5 _2.5 _1  _2.5
)

      m0>0
0 0 1 1 0
0 1 1 1 0
0 1 0 0 0
0 1 0 1 0
0 0 0 0 0

   $z=: 0.4  m0 fm 7   NB. Hurst exponent 0.4
513 513

   $x=: y=: _4+8*(i.%<:)#z
513

   $xyz=: (x ([,])"0/ y),"1 0 z
513 513 3

   quad=: 0 1 3 2&{@(,/)

   $polys=: ,/,/2 2 quad ;._3 xyz
262144 4 3

   range=: (>./-<./)@,

   $rangez=: 2 2 range;._3 {:"1 xyz
512 512

   ]fmc=: 0.1*3 2 0.2,0 2 0,1 4 0,2 2 2,:6
0.3 0.2 0.02
  0 0.2    0
0.1 0.4    0
0.2 0.2  0.2
0.6 0.6  0.6

   $FMC=: interp^:4 fmc
65 3

   $colors=: ,/FMC{~ 65 cile rangez
262144 3

   pfn=: povpath,'fm.pov'

   view_pars_fm fwrites pfn
653

(colors fmtquad polys) fappends pfn
```

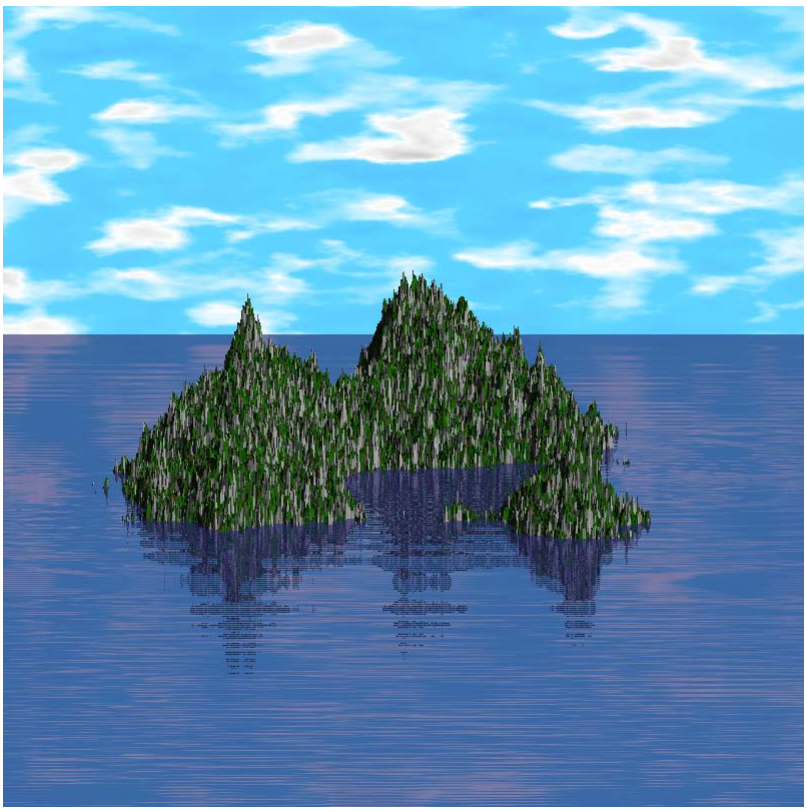


Figure 6. A Fractal Mountain

3. BEST ANALOGS: FRACTAL FORECASTING

Below we create some Henon map data that is chaotic but also has patterns that nearly repeat. The first 80 points are shown in Figure 7. Below we use 900 Henon data points as a reference set and then take a pair 50 points beyond the reference data as a target to forecast from. We use a linear model based upon the closest 5 data points from the reference to do the forecast. The process is repeated for the subsequent pair, which includes the previous forecasted data. Figure 8 shows that the data fits the forecast very well for several steps.

```
hen=: {:, 1:+0.3&*@{.+_1.4&*@*:@{:

   hen^:(i.6) 0.1 0.5
     0.1      0.5
     0.5     0.68
    0.68  0.50264
 0.50264 0.850294
0.850294 0.138592
0.138592   1.2282

   $data=:{.|:hen^:(3+i.1000) 0.1 0.5
1000

   opts=:'labelfont Arial 16;pensize 2;itemcolor 0 255 128'

   opts plot 80{.data
```

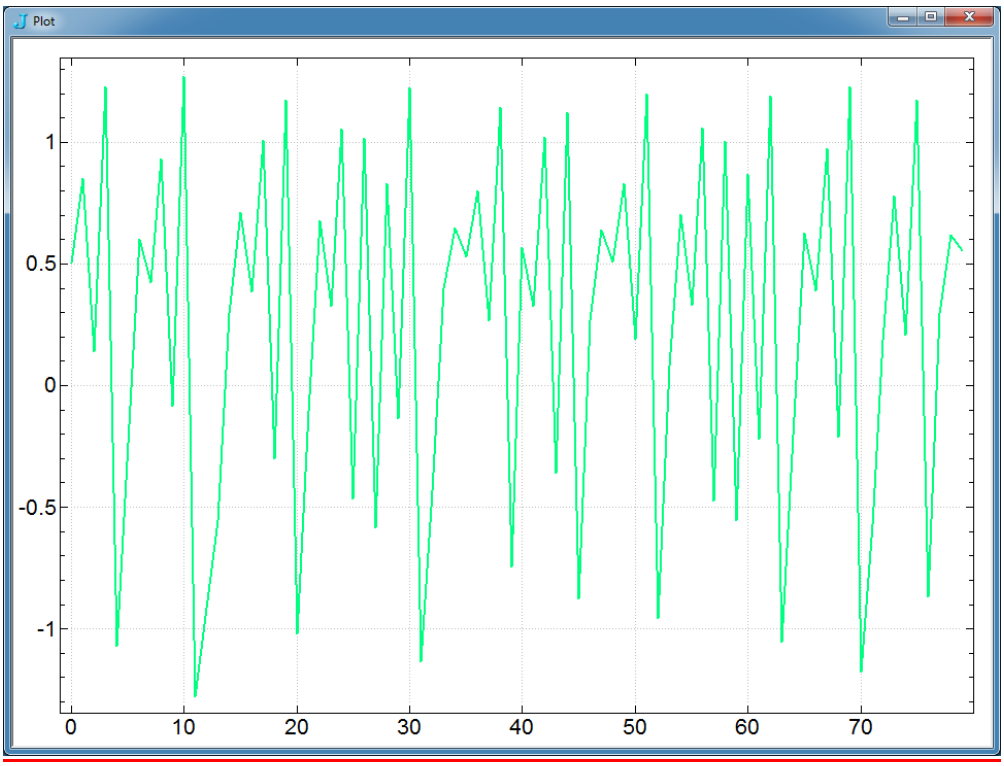


Figure 7. Henon data is chaotic but has some patterns

```
   dist=: +/&.:*:@:-"1

   4 1 dist 2 2
2.23607
   w=:2    NB. window size

   $ref =:}: w ]\ 900{.data
898 2

   _3{.ref
_0.353521 0.574497
 0.574497 0.431878
 0.431878 0.911223

   5{.ref
 0.50264  0.850294
0.850294  0.138592
0.138592    1.2282
  1.2282  _1.07028
_1.07028 _0.235238

   ]sam=:data {~ 950+i.2
1.09422 _0.794183

   5{.sam dist ref
1.74765 0.964141 2.23679 0.306888 2.2355

   5{./:~ sam dist ref
0.00173705 0.00854398 0.0208798 0.0283261 0.031843

   ]j=:5{./: sam dist ref
763 292 497 84 470

   ]coef=:((w+j){data)%. 1,.j{ref
_2.05754 5.19404 4.00548
```

```
   coef +/ . * 1,sam
0.444781

   952{data
0.445247

   ffcast=: 4 : 0
'yy sam'=. y
w=.#sam
'k n'=. x
ref=.}:w ]\ yy
get_j=.k"_ {. [: /: ref"_ dist ]
get_coef=.({&yy@:(w&+) %. 1:,. {&ref)@:get_j
{:|:(}.,get_coef +/ . * 1:,])^:(1+i.n) sam
)

   5 4 ffcast (900{.data);sam
0.444781 0.484614 0.804691 0.238852

   $pred=: 5 20 ffcast (900{.data);sam
20
   $real=: (952+i.20){data
20
   pred,:real
0.444781 0.484614 0.804691 0.238852 1.16152 _0.817137 0.413573  0.51514 0.752282
0.361211 1.04258 _0.413322 1.07369  _0.7382  0.557109 0.343542 1.00194 _0.302616
1.17269 _1.01616
0.445247 0.484202 0.805342 0.237254  1.1628  _0.82176 0.403433 0.525611 0.734257
0.402897 0.99302 _0.259656 1.20352 _1.10573 _0.350637 0.496156 0.55017  0.725086
0.429002 0.959866

   pd 'reset;'
   pd opts
   pd 'type line'
   pd 'pensize 6'
   pd 'itemcolor 0 255 128'
   pd  real
   pd 'itemcolor 0 0 0'
   pd 'pensize 3'
   pd  pred
   pd 'show'
```

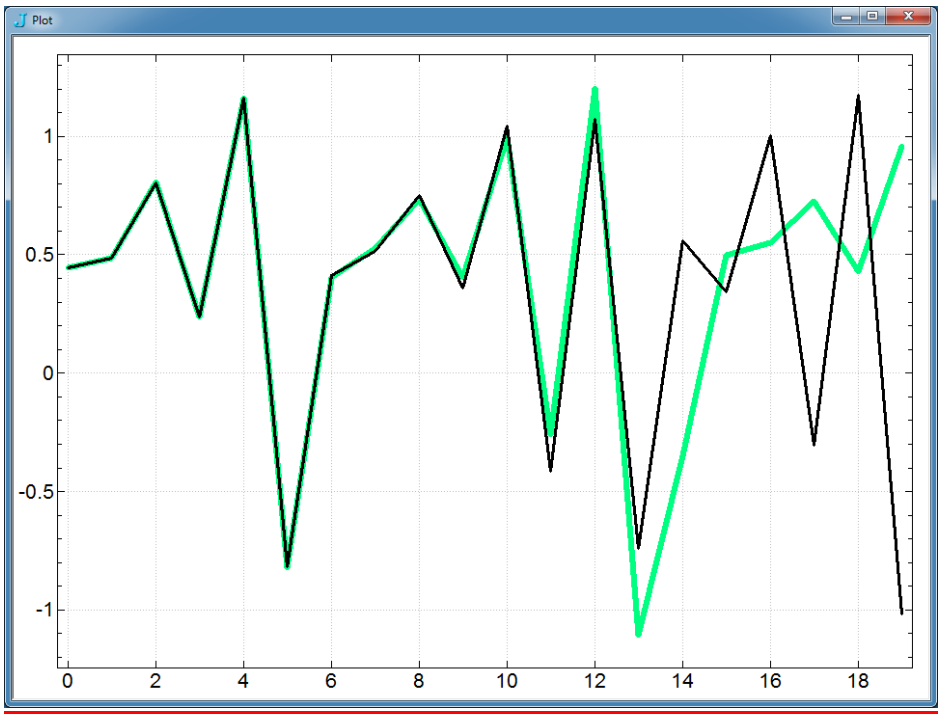


Figure 8. Data and Fractal Forecast

If we can forecast in 1D we can forecast in 2D. Doing similar “forecasting” with missing image data can lead to realistic fake scenery. Figure 9 gives an illustration. Pixels near the edge with missing data are compared to complete patches in the interior and used to update the image. The result is fake but realistic foliage, horizons, clouds, sky and rocks. See “best analogs for missing image data” lab or the link.

http://webbox.lafayette.edu/~reiterc/mvg/bafrmid/index.html

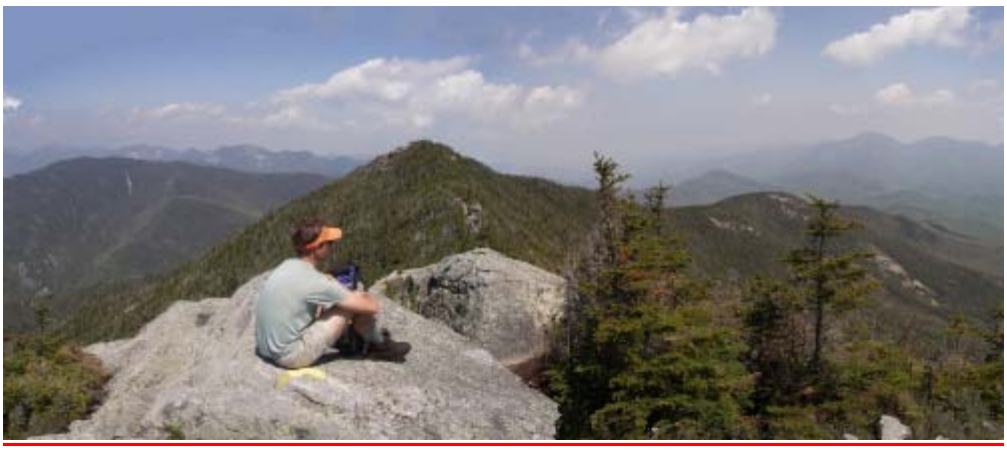


Figure 8. Best Analogs for Missing Image

4. Cellular Automata

Cellular automata are attractive because it seems sensible that the dominat part of behavior in space ought to depend primarily upon local interactions. First we show a cellular automaton that produces patches from a random initial configuration. And then show how to use a larger neighborhood. Some experiments for the reader to try follow.

```
load '~addons\graphics\fvj3\automata.ijs'

lmajor=: (+/ > -:@#)@:,

]n=: 1 1 0,1 0 0,:0 1 1

lmajor n

major=: 3 3&(lmajor;._3)@ perext2

VRAWH=:600 600

20 0.2 major show_auto ?.200 200$2

nperext=: 1 : '(-m)&{. , ] , m&{.'

nperext2=: 1 : '({:m) nperext"1@:(({.m) nperext)'

major2=:5 5&(lmajor;._3)@(2 nperext2)

90 0.15 major2 show_auto ?.200 200$2
```

Figure 9. Majority Rule Organizes Patches

We next describe a simple cellular automaton on a hexagonal lattice that exhibits dendrite growth. Here ones correspond to ice and entries between 0 and 1 can be imagined to be proportionally saturated air. Receptive cells are non-ice neighbors of ice. Receptive cells at the next generation get a small addition, the x argument, and non-ice cell values are locally averaged to simulate diffusion. In Figure 10 we see that dendrite growth evolves.

```
    avg=:+/ % #

cryst=: 3 : 0
0.001 cryst y
:
r=. +./"(1) 1=y,.hxN{y
nrv=.y*-.r
1 <. (avg"1 nrv,.hxN{nrv)+r*y+x
)

    hx_init 230

    800 0 cryst show_hx_auto 1 hxcen}hxA+0.45
```

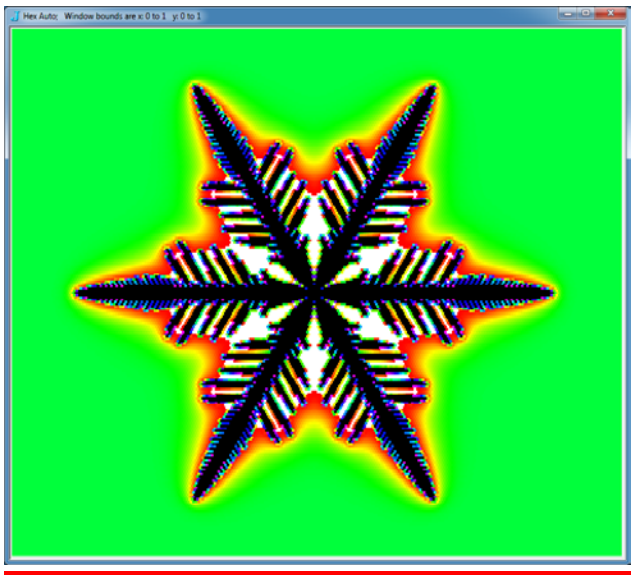


Figure 10. Dendrite Growth

Other values of initial background saturation and addition amount lead to other shapes. See galleries illustrating the variation of those two parameters and the behavior on alternate lattices.

http://webbox.lafayette.edu/~reiterc/mvp/sfn/index.html

http://webbox.lafayette.edu/~reiterc/mvp/qcgm/index.html

This model led to physicists revisiting this problem. They came up with stunning, realistic simulated snowflake growth but used dozens of parameters.

Another favorite type of cellular automata are the Cyclic Cellular Automata where a cell in state k out of possible states 0 to n-1 goes to state k+1 if it has a neighbor in state k+1, otherwise it remains unchanged. It is cyclic in the sense that state 0 is one higher than state n-1. These self organize spirals on a variety of lattices and in 3D.

http://webbox.lafayette.edu/~reiterc/mvq/mscca/index.html

http://webbox.lafayette.edu/~reiterc/mvq/cca_3d/index.html

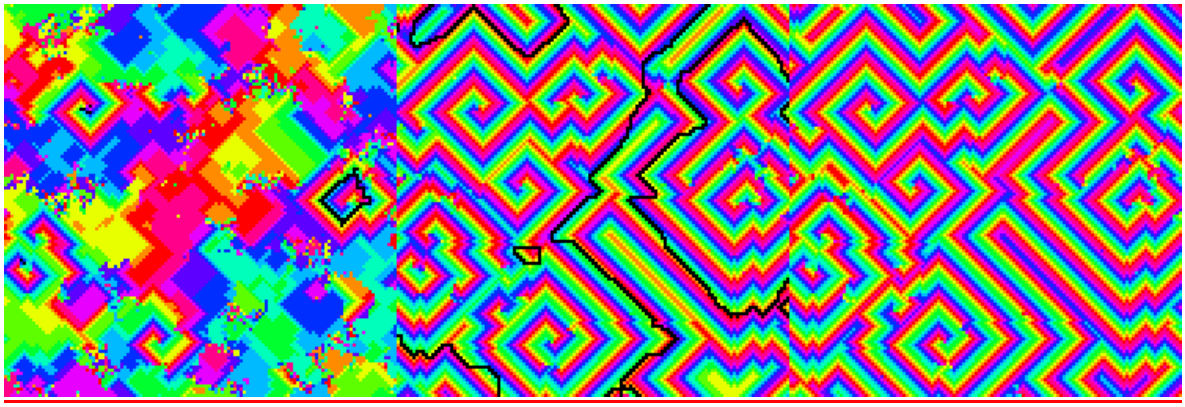


Figure 11. A Cyclic Cellular Automaton Self Organizing into Spirals

Paper:

Probabilistic Prediction

Bo Jacoby

Abstract

Prediction estimates one part from another.

Induction estimates the whole from a part.

Deduction estimates a part from the whole.

These types of problems are common. The traditional statistical approach is to define a plethora of approximate probability distributions (binomial, multinomial, negative binomial, poisson, beta, gamma, dirichlet, wishart etc.). The method described here include all of these distributions as limiting cases when the numbers are big, and when the numbers are small the new method works where traditional methods fail.

These *J*-programs express the *solution* to the above problems.

```
predict=: ([ induct +/@:[ + ]) - [ ,: 0"1
induct =: (T@}: , }.)@(T deduct (T=.(-@+)#)~)
deduct =: *`%`:3"2 @ (,: (%:@* -.)) @ ((,: , 1"1) % +/@[)
```

While the deduction distribution is the well known multivariate hypergeometric distribution, the induction and prediction distributions are surprisingly not *conspicuous* in the litterature. Please tell me if you find it.

This can potentionally *revolutionize* statistical education and practice.

Concepts

Prediction is not generally exact, but a number can be estimated by an *order of magnitude*, give or take an *uncertainty*.

number ≈ order of magnitude ± uncertainty

The situation is modelled by a jar containing colored beads. The *whole* is the contents of the jar. A *part* is a sample taken from the jar.

Deduction: Estimate the number of beads of different colors in the *sample* from the number of beads of different colors in the jar, and the total number of beads in the sample.

Induction: Estimate the number of beads of different colors in the *jar* from the number of beads of different colors in the sample, and the total number of beads in the jar.

Prediction: Estimate the number of beads of different colors *outside* the sample from the number of beads of different colors in the sample, and the total number of beads outside the sample.

This is expressed symbolically like this.

$K = (K_1, \ldots, K_I)$ = the number of beads of different colors in the jar.

$k = (k_1, \ldots, k_I)$ = the number of beads of different colors in the sample.

$K - k$ = the number of beads of different colors outside the sample.

$k \approx K$ `deduct` $\sum k$

$K \approx k$ `induct` $\sum K$

$K - k \approx k$ `predict` $\sum (K - k)$

1 Examples

1.1 Prediction

One sample consist of zero white and one black bead. The color distribution of the remaining 3 beads is to be estimated. This is written like this.

```
0 1 predict 3
```

The answer is two lines.

```
1 2
1 1
```

The first line is the *orders of magnitude* of white and black beads. The second line is the *uncertainties*. So the result is (1, 2)±(1, 1) = (1±1, 2±1). The remaining beads are 1±1 white beads and 2±1 black beads.

The program reproduces the trivial result:

```
1 1 predict 0
0 0
0 0
```

Without observation all possibilities are equally likely.

```
0 0 predict 1
0.5 0.5
0.5 0.5

0 0 0 predict 3
1 1 1
1 1 1

0 0 predict 6
3 3
2 2
```

Now decipher these examples.

```
0 1 predict 24
8 16
6  6

3 3 3 predict 300
100 100 100
 40  40  40

1000000000 1000000000 predict 4
2 2
1 1
```

```
        1000000000 1000000000 predict 100
     50 50
      5  5
```

1.2 Induction

There are 6 white beads and 4 black beads in a sample from a jar containing 100 beads.

```
        6 4 induct 100
        58.5    41.5
     13.1009 13.1009
```

The whole jar contained 59±13 white beads and 41±13 black beads.

This sample is the whole jar.

```
        6 4 induct 10
     6 4
     0 0
```

The uncertainties are zero.

Here all the beads in the sample have the same color.

```
        10 0 induct 100
        92.5     7.5
     7.34454 7.34454
```

The same is not necessarily true for the whole jar. The uncertainties are nonzero.

There are 6 white, 4 black, and 0 red beads in a sample from a jar containing 100 beads.

```
        6 4 0 induct 100
     54.4615 38.6154 6.92308
     12.8279 12.5188 6.85683
```

The whole jar contained 54±13 white, 39±13 black, and 7±7 red beads. The jar may have contained red beads even if the sample did not.

If there are lots of beads in the jar then it will have an approximate Dirichlet distribution

```
        6 4 0 induct 1000000
     538462 384616 76922.3
     133234 130023 71216.7
```

(54±13)% white, (38±13)% black, and (8±7)% red.

Bigger sample makes smaller standard deviation.

```
        60 40 0 induct 1000000
      592234  398058 9707.77
     48185.3 47996.9 9614.47
```

1.3 Deduction

From a jar with 60 white and 40 black beads, 10 beads are taken

```
        60 40 deduct 10
          6      4
     1.4771 1.4771
```

There will be 6.00±1.48 white and 4.00±1.48 black beads in the sample.

From a jar with 60 white, and 30 black, and 10 red beads, 10 beads are to be taken.

```
         10 deduct 60 30 10
           6      3        1
      1.4771 1.3817 0.904534
```

There will be 6.00±1.48 white and 3.00±1.38 black and 1.00±0.90 red beads in the sample.

If the sample is the whole jar, then the uncertainties are zero.

```
         60 30 10 deduct 100
      60 30 10
       0  0  0
```

All the beads in the jar have the same color.

```
         100 0 deduct 10
      10 0
       0 0
```

The same is the case for the sample. This is *logical* deduction. There is no uncertainty.

If there are lots of beads in the jar then small samples have binomial distributions.

```
         60000000 40000000 deduct 10
            6       4
      1.54919 1.54919
```

or multinomial distributions

```
         60000000 30000000 10000000 deduct 10
            6       3        1
      1.54919 1.44914 0.948683
```

If there are relatively few white beads in the jar then the number of whie beads in the sample will have a Poisson distribution.

```
         1000000 999000000 deduct 4000
          4  3996
      1.999 1.999
```

So the formulas for the Binomial and Poisson distributions are special cases of the Deduction Formula.

2 Formulas

2.1 Deduction

Estimate k from $(K, \sum k)$.

$$k \approx \frac{\frac{\sum k\, K}{\sum K \sum K}}{\frac{1}{\sum K}} \pm \frac{\sqrt{\frac{\sum k}{\sum K}\left(1-\frac{\sum k}{\sum K}\right)}\sqrt{\frac{K}{\sum K}\left(1-\frac{K}{\sum K}\right)}}{\sqrt{\frac{1}{\sum K}\left(1-\frac{1}{\sum K}\right)}}$$

This is computed in 3 steps:

1. $(a_{11i}, a_{12i}, a_{13i}) = \frac{(\sum k, K_i, 1)}{\sum K}$ for $i = 1, \ldots, I$
2. $a_{2hi} = \sqrt{a_{1hi}(1 - a_{1hi})}$ for $h = 1, 2, 3$ and $i = 1, \ldots, I$
3. $b_{ji} = \frac{a_{j1i} a_{j3i}}{a_{j2i}}$ for $j = 1, 2$ and $i = 1, \ldots, I$

Then $k_i \approx b_{1i} \pm b_{2i}$ for $i = 1, \ldots, I$

2.2 Induction

Estimate K from $(k, \sum K)$.

$$K \approx -1 - \frac{\frac{\sum(-1-K)}{\sum(-1-k)}\frac{-1-k}{\sum(-1-k)}}{\frac{1}{\sum(-1-k)}} \pm \frac{\sqrt{\frac{\sum(-1-K)}{\sum(-1-k)}\left(1-\frac{\sum(-1-K)}{\sum(-1-k)}\right)}\sqrt{\frac{-1-k}{\sum(-1-k)}\left(1-\frac{-1-k}{\sum(-1-k)}\right)}}{\sqrt{\frac{1}{\sum(-1-k)}\left(1-\frac{1}{\sum(-1-k)}\right)}}$$

(where $\sum(-1-K) = -(\sum k^0 - \sum K) = -I - \sum K$ is known.)

This is computed in 3 steps:

1. replace $(K, \sum k)$ by $(-1-k, \sum(-1-K))$
2. estimate $k \approx \mu_k \pm \sigma_k$ by the Deduction Formula
3. compute $K = -1-k \approx (-1-\mu_k) \pm \sigma_k$

2.3 Prediction

Estimate $K-k$ from $(k, \sum(K-k))$.

$$K-k \approx -1-k - \frac{\frac{\sum(-1-K)}{\sum(-1-k)}\frac{-1-k}{\sum(-1-k)}}{\frac{1}{\sum(-1-k)}} \pm \frac{\sqrt{\frac{\sum(-1-K)}{\sum(-1-k)}\left(1-\frac{-1-K}{\sum(-1-k)}\right)}\sqrt{\frac{-1-k}{\sum(-1-k)}\left(1-\frac{-1-k}{\sum(-1-k)}\right)}}{\sqrt{\frac{1}{\sum(-1-k)}\left(1-\frac{1}{\sum(-1-k)}\right)}}$$

This is computed in 3 steps:

1. let $\sum K = \sum(K-k) + \sum k$
2. estimate $K \approx \mu_K \pm \sigma_K$ by the Induction Formula
3. compute $K-k \approx (\mu_K - k) \pm \sigma_K$

3 Proofs

3.1 Lemmas

If a formula is proved in the book *Concrete Mathematics.*[1] the proof is not repeated here.

---

[1] Graham, Ronald L.; Knuth, Donald E.; Patashnik, Oren (1994). *Concrete Mathematics (Second ed.). Reading, MA: Addison-Wesley Professional. pp. xiv+657.* ISBN 0-201-55802-5. MR 1397498.

The number of *n*-sized parts from a *N*-sized jar is the *binomial coefficient* (N), which is computed recursively by the initial condition $(N) = 1$, Conc.Math 5.2, together with the *absorption identity* $n(N) = N\binom{N-1}{n-1}$, Conc.Math. 5.6. The absorption identity is easily generalized to the following two formulas, which are written side by side in order to emphasize a strange symmetry which leads to the symmetry between deduction and induction.

$(n)(N) = (N)\binom{N-J}{n-J}$	$(-1-N)(N) = (-1-n)\binom{N+J}{n+J}$

for $J = 0, 1, 2, \ldots$

The *negation formula* $(N) = (-1)^n(n-1-N)$ is Conc Math 5.14.

The *binomial formula*, Conc.Math. 5.13, and the *negative binomial formula* are

$(1+x)^N = \sum_n (N)x^n$	$(-1+x)^{-1-n} = \sum_N (N)x^{-1-N}$

To the left the summation is over n and to the right the summation is over N .

Proof of negative binomial formula.

$\sum_N (N)x^{-1-N}$	(right hand side)
$=\sum_N (N+n)x^{-1-(N+n)}$	(replace index N by $N+n$)
$=\sum_N (-1)^N(N-1-(N+n))x^{-1-n}x^{-N}$	(negation formula)
$=x^{-1-n}\sum_N (-1-n)(-x^{-1})^N$	(constant factor outside summation)
$=x^{-1-n}(1+(-x^{-1}))^{-1-n}$	(binomial formula)
$=(x(1+(-x^{-1})))^{-1-n}$	$(a^n b^n = (ab)^n)$
$=(-1+x)^{-1-n}$	(= left hand side)

The number of *k*-distributed parts from a *K*-distributed jar is the product $\prod\binom{K}{k} = \binom{K_1}{k_1}\ldots\binom{K_I}{k_I}$

The *Iverson bracket* notation is [**true**]=1 and [**false**]=0.

The following formulas for $J = 0, 1, 2, \ldots$ are used to simplify sums of product of binomial coefficients.

$\sum_k [\Sigma k = n]\binom{k}{J}\prod\binom{K}{k}$	$\sum_K [\Sigma K = N]\binom{-1-K}{J}\prod\binom{K}{k}$
$=\binom{K}{J}\binom{-J+\Sigma K}{-J+\Sigma k}$	$=\binom{-1-k}{J}\binom{-1+J-\Sigma(-1-K)}{-1+J-\Sigma(-1-k)}$

The special case ($J = 0$, $I = 2$) is Conc.Math. 5.22 and 5.26.

The case $J = 0$ is proved by showing that the *generating function* (see Conc.Math. 7) of the left hand side is equal to the generating function of the right hand side. The intermediate results are shaded.

$$\sum_n \left(\sum_k [\Sigma k = n]\prod(K)\right)x^n$$
$$=\prod\Sigma_k \ (\Sigma_n \ [\Sigma k = n])(K)x^k$$
$$=\prod\Sigma_k \ (K)x^k$$
$$=\prod(1+x)^K$$
$$=(1+x)^{\Sigma K}$$
$$=\Sigma_n \ (\Sigma K)x^n$$
$$=\Sigma_n \ (\Sigma K)x^n$$

$$\sum_N \left(\sum_K [\Sigma K = N]\prod(K)\right)x^{-(I+N)}$$
$$=\prod\Sigma_K \ (\Sigma_N \ [\Sigma K = N])(K)x^{-1-K}$$
$$=\prod\Sigma_K \ (K)x^{-1-K}$$
$$=\prod(-1+x)^{-1-k}$$
$$=(-1+x)^{\Sigma(-1-k)}$$
$$=\Sigma_N \ \binom{N}{-1-\Sigma(-1-k)}x^{-1-N}$$
$$=\Sigma_N \ \binom{-1-(-I-N)}{-1-\Sigma(-1-k)}x^{-I-N}$$

The case for $J > 0$ is proved like this

$$\sum_k [\Sigma k = n]\binom{k}{J}\prod\binom{K}{k}$$
$$=\Sigma_k \ [\Sigma k = n]\frac{(k_i)\binom{K_i}{k_i}}{\binom{K_i}{k_i}}\prod(K)$$
$$=\Sigma_k \ [\Sigma k = n]\frac{(K_i)\binom{-J+K_i}{-J+k_i}}{\binom{K_i}{k_i}}\prod(K)$$
$$=\binom{K}{J}\binom{-J+\Sigma K}{-J+\Sigma k}$$

$$\sum_K [\Sigma K = N]\binom{-1-K}{J}\prod\binom{K}{k}$$
$$=\Sigma_K \ [\Sigma K = N]\frac{(1+K_i)\binom{K_i}{k_i}}{\binom{K_i}{k_i}}\prod(K)$$
$$=\Sigma_K \ [\Sigma K = N]\frac{\binom{-1-k_i}{J}\binom{-1+J+K_i}{-1+J+k_i}}{\binom{K_i}{k_i}}\prod(K)$$
$$=\binom{-1-k}{J}\binom{-1+J-\Sigma(-1-K)}{-1+J-\Sigma(-1-k)}$$

Fractions between sums of products of binomial coefficients are simple rational expressions.

$$J\frac{\Sigma_k \ [\Sigma k = n]\binom{k}{J}\prod\binom{K}{k}}{\Sigma_k \ [\Sigma k = n]\binom{k}{-1+J}\prod\binom{K}{k}}$$
$$=J\frac{\binom{K}{J}}{\binom{K}{-1+J}}\frac{\binom{-J+\Sigma K}{-J+\Sigma k}}{\binom{-(-1+J)+\Sigma K}{-(-1+J)+\Sigma k}}$$
$$=K\frac{1-J+\Sigma k}{1-J+\Sigma K}$$

$$J\frac{\Sigma_K \ [\Sigma K = N]\binom{-1-K}{J}\prod\binom{K}{k}}{\Sigma_K \ [\Sigma K = N]\binom{-1-K}{-1+J}\prod\binom{K}{k}}$$
$$=J\frac{\binom{-1-k}{J}}{\binom{-1-k}{-1+J}}\frac{\binom{-1+J-\Sigma(-1-K)}{-1+J-\Sigma(-1-k)}}{\binom{-1+(-1+J)-\Sigma(-1-K)}{-1+(-1+J)-\Sigma(-1-k)}}$$
$$=(-1-k)\frac{1-J+\Sigma(-1-K)}{1-J+\Sigma(-1-k)}$$

We need the cases J=1 and J=2:

$$\frac{\Sigma_k \ [\Sigma k = n]k\prod\binom{K}{k}}{\Sigma_k \ [\Sigma k = n]\prod\binom{K}{k}}$$
$$=K\frac{\Sigma k}{\Sigma K}$$

$$\frac{\Sigma_K \ [\Sigma K = N](-1-K)\prod\binom{K}{k}}{\Sigma_K \ [\Sigma K = N]\prod\binom{K}{k}}$$
$$=(-1-k)\frac{\Sigma(-1-K)}{\Sigma(-1-k)}$$

$\frac{\sum_k \ [\sum k = n] k(k-1)\prod\binom{K}{k}}{\sum_k \ [\sum k = n] k\prod\binom{K}{k}}$ $= K\frac{-1+\sum k}{-1+\sum K}$	$\frac{\sum_K \ [\sum K = N](-1-K)(-2-K)\prod\binom{K}{k}}{\sum_K \ [\sum K = N](-1-K)\prod\binom{K}{k}}$ $= (-1-k)\frac{-1+\sum(-1-K)}{-1+\sum(-1-k)}$

The left column is used in proving the deduction formula, and the right column is used in proving the induction formula.

3.2 Deduction

The deduction estimate is $k \approx \mu_k \pm \sigma_k$.

The orders of magnitude μ_k and uncertainties σ_k are defined by

$$\mu_k = \frac{\sum_k \ [\sum k=n] k\prod(K)}{\sum_k \ [\sum k=n]\prod(K)} \text{and} \mu_k^2 + \sigma_k^2 = \frac{\sum_k \ [\sum k=n] k^2\prod(K)}{\sum_k \ [\sum k=n]\prod(K)}$$

To speed up computation the expressions are simplified.

A: $\mu_k = \frac{K\sum k}{\sum K}$

B: $-1 + \mu_k + \frac{\sigma_k^2}{\mu_k} = \frac{(-1+K)(-1+\sum k)}{-1+\sum K}$

From A and B it follows that

$$\mu_k = \frac{\frac{\sum k \ K}{\sum K \sum K}}{\frac{1}{\sum K}}$$

and

$$\frac{\sigma_k^2}{\mu_k} = 1 - \frac{K\sum k}{\sum K} + \frac{(-1+K)(-1+\sum k)}{-1+\sum K} = \frac{((\sum K)-\sum k)((\sum K)-K)}{(-1+\sum K)\sum K}$$

$$= \frac{\left(1-\frac{\sum k}{\sum K}\right)\left(1-\frac{K}{\sum K}\right)}{\left(1-\frac{1}{\sum K}\right)}$$

which implies the deduction formula. *Q.E.D.*

3.3 Induction

The induction estimate is $K \approx \mu_K \pm \sigma_K$.

The orders of magnitude μ_K and uncertainties σ_K are defined by

$$\mu_K = \frac{\sum_K \ [\sum K=N] K\prod(K)}{\sum_K \ [\sum K=N]\prod(K)} \text{and} \mu_K^2 + \sigma_K^2 = \frac{\sum_K \ [\sum K=N] K^2\prod(K)}{\sum_K \ [\sum K=N]\prod(K)}.$$

Using $\mu_{-1-K} = -1 - \mu_K$ and $\sigma_{-1-K} = \sigma_K$ get

A: $\mu_{-1-K} = \frac{(-1-k)\sum(-1-K)}{\sum(-1-k)}$

B: $$-1+\mu_{-1-K}+\frac{\sigma^2_{-1-K}}{\mu_{-1-K}}=\frac{(-1+(-1-k))(-1+\sum(-1-K))}{-1+\sum(-1-k)}$$

Replacing (k, K) by $(-1-K, -1-k)$ in the deduction formulas A and B, the induction formulas A and B are reproduced. *Q.E.D.*

3.4 Prediction

The induction formula implies the prediction formula in a trivial way. The two samples together constitute the jar, and inference from one sample to the jar is done by the induction formula. The other sample is obtained by subtracting the first sample from the jar. The uncertainty of the other sample is the same as the uncertainty of the jar.

Elegant Expressions

David Alis

Because the J clipboard interface is so convenient I often use it to manipulate text. So when I needed to document a few thousand lines of Excel VBA code I used it to help me. Using J provided a welcome break from the tedium of Visual Basic and an opportunity to admire the elegance of Ken Iverson's thinking. This time it was a particular application of the Under conjunction in the expression "`E.&.|.`" (MemberOfInterval Under Reverse). This simple fragment was an epiphany. In this neat, tiny and elegant expression I realised that elegance and balance is the reason for my continuing interest in J and APL. This brief note presents four other elegant expressions.

`E.&.|.` `MemberOfInterval Under Reverse`

For my Excel VBA problem I needed to analyse the text of a code module and arrange the functions into alphabetical order. When you use the VBA code editor the list box selector offers the functions alphabetically but in the module itself they are in the haphazard order in which they were entered.

My code modules are collections of either a Sub or a Function and which terminate with an '`End Sub`' or an '`End Function`' statement.

```
VBA =: 0 : 0
Sub test()
Print “line test”
End Sub
Sub retest()
Print “line retest”
End Sub
```

The primitive `E.` (`MemberOfInterval`) was the natural choice. But using `E.` in the expression `'End Sub' E.` VBA returns a boolean vector where 1's locate the start of 'End Sub' not the end. By reversing the string VBA and using `'buS dnE'` as the search string the 1's are located correctly but the boolean vector must be reversed.

```
|.'buS dnE' E. |.VBA
```

Which leads immediately to the marvellous expression

```
'End Sub' E. &.|.VBA
```

I have been using Under "`&.`" for many years but this was first time I had stopped and reflected on the type of thinking needed to design the languages.

} `ItemAmend`

ItemAmend is seldom seen in the library scripts. I don't know why.

It gets my vote in the beauty contest.

```
 p =: 8 0 9 12 8 3 8

 (p=8)} p,:12345

 12345 0 9 12 12345 3 12345
```

```
     ] p =:  3 3$'one';'two';'three';'four';'five';'one';'nine';'one'
+----+----+-----+
|one |two |three|
+----+----+-----+
|four|five|one  |
+----+----+-----+
|nine|one |one  |
+----+----+-----+

    (p=<'one') } p,:<'hello sailor'
+------------+------------+------------+
|hello sailor|two         |three       |
+------------+------------+------------+
|four        |five        |hello sailor|
+------------+------------+------------+
|nine        |hello sailor|hello sailor|
+------------+------------+------------+
```

`leapyear`

`0 ((~:/) . =) (4 100 400 (|/) ])`

This expression tests for leap years and uses the residue verb table, **`(|/)`** and the generalised dot product of `not-equals reduction` and `equals` **`((~:/) . =)`**.

`leapyear` is an anonymous fork function which is itself composed of inner anonymous functions. While this is not beginner stuff most people seeing the 4 100 400 could guess that the algorithm had something to do with leap years.

The Representation section of J User Guide mentions symbol-less entities in the description of 5!:1. But I think anonymous is a better description and it also sounds less intimidating. Anonymous functions are common in Javascript, Java, Perl and other languages. In these languages anonymous means that the defined function doesn't have a proper name. Normally such anonymous functions could not be called recursively.

Two of these inner functions stand for the constant verbs `0"_` and `(4 100 400)"_ .` The inner anonymous function **`(4 100 400 (|/) ])`** is a fork around the residue verb table.

The dot product in the centre of `leapyear` is evaluated by the expression **`0 (~:/) . = A`** and because the left argument is a scalar it is the same as **`~:/ 0 = A.`**

```
    [ B =: 1900, 1995 + i.10
1900 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004

    [ A =: (4 100 400 (|/) ]) B
  0   3   0   1   2   3 0 1 2 3 0
  0  95  96  97  98  99 0 1 2 3 4
300 395 396 397 398 399 0 1 2 3 4

    0 = A
1 0 1 0 0 0 1 0 0 0 1
1 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0

    ~:/ 0 = A
0 0 1 0 0 0 1 0 0 0 1
```

```
+-----------------------------------------------------+
|B , A                                                |
|1900 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004|
|   0    3    0    1    2    3    0    1    2    3    0|
|   0   95   96   97   98   99    0    1    2    3    4|
```

```
| 300  395  396  397  398  399    0    1    2    3    4|
+-------------------------------------------------------+
|B , 0=A                                                |
|1900 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004|
|   1    0    1    0    0    0    1    0    0    0    1|
|   1    0    0    0    0    0    1    0    0    0    0|
|   0    0    0    0    0    0    1    0    0    0    0|
+-------------------------------------------------------+
|B,:~:/ 0 =A                                            |
|1900 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004|
|   0    0    1    0    0    0    1    0    0    0    1|
+-------------------------------------------------------+
```

The rule states that if a year is divisible by four then its leap unless its divisible by 100 then it's not but when it's divisible by 400 then it is is – which is expressed by `not-equals reduction`

```
   ~:/ 0 0 0     NB. 1995

0

   ~:/ 1 0 0     NB. 1996

1

   ~:/ 1 1 0     NB. 1900

0

   ~:/ 1 1 1     NB. 2000

1
```

`#.` Base and `#:` Antibase

The definition of base says it's a weighted sum but the Vocabulary examples don't really reveal how clever base and antibase are.

There are 60 seconds in a minute, 60 minutes in an hour, 24 hours in a day and 7 days in a week. To show that there nearly 31.5 million seconds in a 365 day year, i.e. fifty two weeks and one day:

```
  0  7 24 60 60 #. 0 365 0 0 0,: 52 1 0 0 0
31536000 31536000
```

According to Google there are 35.274 ounces per kilogram. I weigh 90Kg instead of 80Kg meaning that I'm 10 Kg overweight. But I grew up in the UK before it adopted the metric scale so I still need to be told this stones and pounds. There are 16 ounces in a pound and 14 pounds in a stone. Apparently I'm almost a stone and half overweight.

```
   0 14 16 #: (35.274 * 90 80)
```

```
14 2 6.66
12 8 5.9
```

Besides the 16 ounces and the 14 pounds there are 8 stone to a hundredweight and 20 hundredweights to an imperial ton. In other words there are 35,840 ounces in a ton or 1,016 Kg (thereabouts).

```
   0 20 8 14 16 #. 0 20 0 0 0,: 1 0 0 0 0
35840 35840

   35840 %  35.274
1016.05
```

Besides the imperial ton there are long tons, short tons and also metric tonnes. A short ton has 32,000 ounces while the long ton and the imperial ton have the same weight. So the short ton is about 907 Kg. Perhaps I should update the Wikipedia entry for Avoirdupois to include some nice worked examples.

; Link

Link has a short definition and includes a few brief examples. It's only a slight exaggeration to say that **Link** appears on every page of the Vocabulary. It succinctly assembles the pieces of an explanation without clutter.

Here is a variation of part of the description of **/: Gradeup** that explains how to fold upper and lower case, i.e. bringing A and a together and before the other letters.

```
      Aa=: ' ',. a. {~ 65 97 +/ i. 26
      x=: words=: >;: 'When eras die'
      j=: <./Aa i."1 _ x
      x ; (x/:x) ; (x/:j) ; Aa

 x ; (x/:x) ; (x/:j);j ;(Aa i."1 _ x); Aa

+----+----+----+----------+----------+--------------------------+
|When|When|die |23  8 5 14|23 27 27 27| ABCDEFGHIJKLMNOPQRSTUVWXYZ|
|eras|die |eras| 5 18 1 19|27 27 27 27| abcdefghijklmnopqrstuvwxyz|
|die |eras|When| 4  9 5  0|27 27 27  0|                          |
|    |    |    |          |          |                          |
|    |    |    |          |27  8  5 14|                          |
|    |    |    |          | 5 18  1 19|                          |
|    |    |    |          | 4  9  5  0|                          |
+----+----+----+----------+----------+--------------------------+
```

```
 }:'words' F  'words/:words' F '(words/:j)' F 'j ' F '(Aa i."1 _ words)' F 'Aa' foo ' '
+-----+------------+----------+----------+-----------------+--------------------------+
|words|words/:words|(words/:j)|j         |(Aa i."1 _ words)|Aa                        |
|     |            |          |          |                 |                          |
|When |When        |die       |23  8 5 14|23 27 27 27      | ABCDEFGHIJKLMNOPQRSTUVWXYZ|
|eras |die         |eras      | 5 18 1 19|27 27 27 27      | abcdefghijklmnopqrstuvwxyz|
|die  |eras        |When      | 4  9 5  0|27 27 27  0      |                          |
|     |            |          |          |                 |                          |
|     |            |          |          |27  8  5 14      |                          |
|     |            |          |          | 5 18  1 19      |                          |
|     |            |          |          | 4  9  5  0      |                          |
|     |            |          |          |                 |                          |
+-----+------------+----------+----------+-----------------+--------------------------+

NB. F is a simple wrapper around ; to facilitate the decoration.
F=.4 : ('a=.":".x';'if. 1=#$a do. a=.,:a end.';'if. 2< #$a do. a=. ;<"2 '' '',~"2 a end.';'(x ,'' '',a);y')
```

Short NOTE:

Quotations from *Machine Solutions of Linear Differential Equations*

Roger Hui

Quotations from Iverson, K.E., *Machine Solutions of Linear Differential Equations – Applications to a Dynamic Economic Model*, Doctoral Thesis, Harvard University, 1954-01.

- The writer wishes to express his appreciation to Professor Howard H. Aiken for his helpful guidance and encouragement, and to Professor Wassily W. Leontief who proposed the problem and offered many valuable suggests for its solution.

— Preface

[Howard Aiken (1900-1973) was a major figure of the early digital era. He is best known for his first machine, the IBM Automatic Sequence Controlled Calculator or Harvard Mark I, conceived in 1937 and put into operation in 1944. But he also made significant contributions to the development of applications for the new machines and to the creation of a university curriculum for computer science. (From I. Bernard Cohen, *Howard Aiken: Portrait of a Computer Pioneer*, MIT Press, 1999.)]

[Wassily Leontief won the Nobel Prize in Economics in 1973 “for the development of the input-output method and for its application to

important economic problems". Three of his students also won the prize.]

- The writer also wishes to express his thanks to Miss Jacquelin Sanborn who typed the plates for printing … and to … Mr. Robert Burns … all of whom assisted in the preparation of the manuscript.

  — Preface

  [Jacquelin Sanborn was also acknowledged in *A Programming Language* for providing "much early and continuing guidance in matters of style, format, and typography". Robert Burns was acknowledged in same as among "unusually competent typists and draughtsmen" who provided assistance.]

- The speed of modern computers has reached a point where, for many applications, the time required to program a problem may be a more important factor than the actual computing time. Attempts are made, therefore, to simplify the work of programming in various ways.

  — Section 2A4, Simplification of Programming

- The calculator has a storage capacity of 10,000 orders, each of which is identified by a "line-number" in the range 0000-9999. …

  The two hundred fast storage registers are referred to by the numbers of the β-codes controlling them and are numbered from 0000 to 0199. …

  The function registers are ten in number (β0220-0229) and are designated as f0-9. …

  Provision is made for the storage of four thousand numbers in "slow storage" positions numbered 0000-3999. …

  The average time required for certain operations is given below in terms of machine cycles. A machine cycle is 1.2 milliseconds, the time required for a single order.

  — Section 2B, Mark IV Programming

- All equations are to be understood as matrix or vector relations unless otherwise specified.

  — Section 3, Linear Differential Equations

- Since the matrix is not symmetric, complex roots may be expected to occur and indeed these complex roots are of particular interest in the theory of

the business cycle.
(The term of the general solution corresponding to a pair of complex roots is periodic.)

— Section 4, Latent Roots and Principal Vectors

- Uniformity is desirable in automatic computation since exceptions may require as much programming as the main process.

— Section 5A4, The Quadratic Factor Method

- The Frame method [for generating the characteristic equation] was first run using floating vector operation. This proved satisfactory for the matrices of order ten and less but the round-off accumulation in the higher order systems was too severe. The calculation was repeated, therefore, using double-accuracy operation. However, the approximate value of the solution obtained by the floating vector method was useful in choosing the decimal point in the double-accuracy calculation so as to minimize the round-off error. For a matrix of order twenty the floating vector program takes two hours and the double-accuracy program six hours.

— Section 6D, The Method of Frame

- The machine computations were begun during the final testing of Mark IV when the machine was first coming into operation. Hence conditions were far from ideal and the lack of experience in programming and operation combined with machine failures to cause a considerable loss of time. However it is under just such adverse conditions that the relative ease of programming reruns on Mark IV proves especially valuable. The reliability of the computer has now been greatly improved and at present writing it has been operating with more than 85% good running time during the month of November 1953.

Mathematical Corner:

Szegö Curve and J accuracy

Mikel Paternain

1. INTRODUCTION

In this issue, *The Zeros of the Partial Sums of* e^z, Roger Hui presents an interesting question about mathematics functions and J accuracy. As in the real case, the exponential function can be defined on the complex plane definition parallels the power series definition for real numbers, where the real variable is replaced by a complex one. The mathematical function e^z with z complex admits the next expansion:

$$e^z = \sum_{n=0}^{\infty} \frac{z^n}{n!} \text{ with } z \in \mathbb{C}$$

2. SZEGÖ CURVE

In 1924 Szegö showed that the zeros of the normalized partial sums $s_n(nz)$ of e^z tended to what is now called the *Szegö curve S*, where

$$S := \{z \in \mathbb{C}: |ze^{1-z}| = 1 \; and \; |z| \leq a\}$$

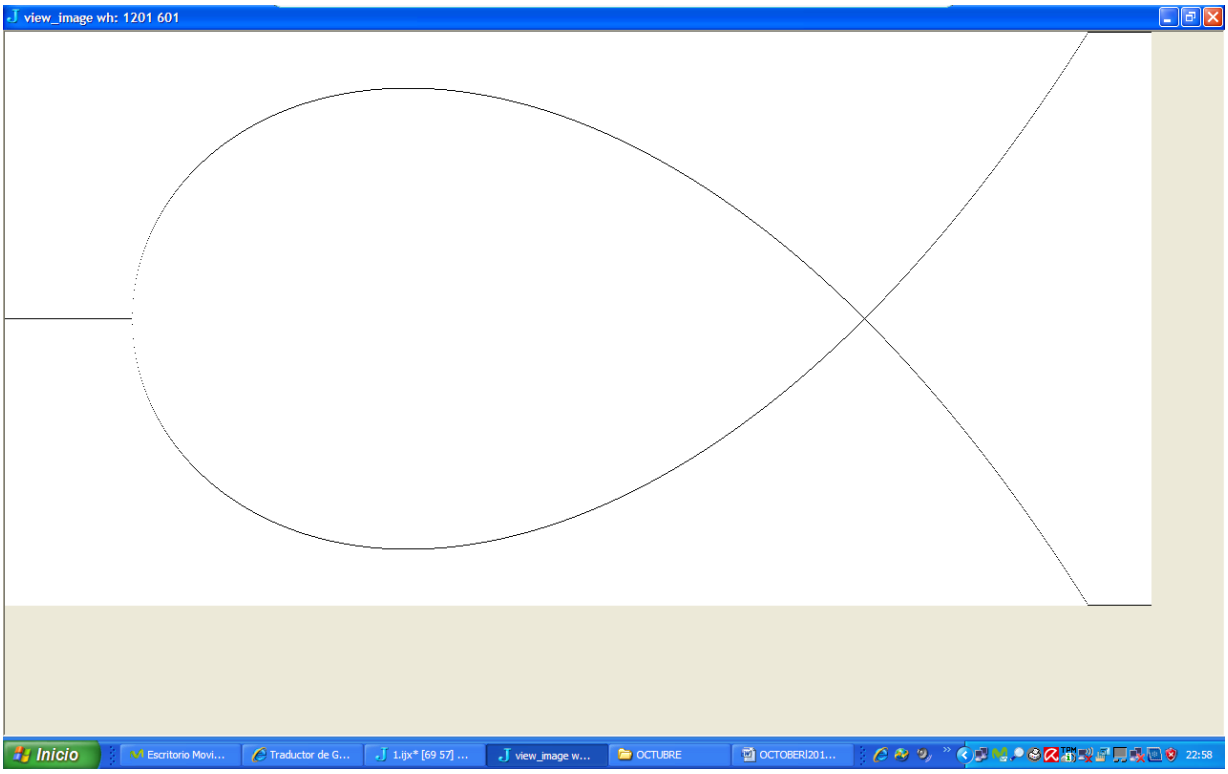
3. J CODE

Thanks to professor Cliff Reiter for J code.

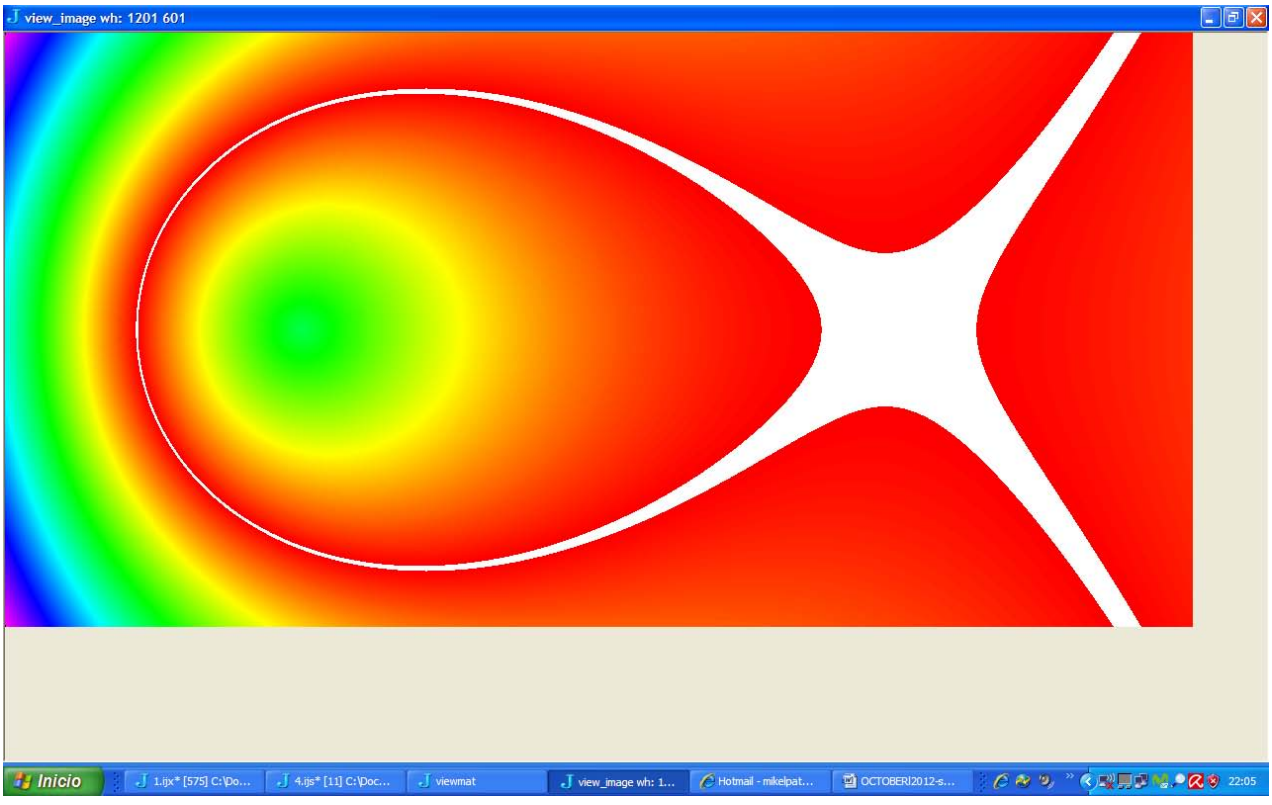
```
zl_clur=: 4 : 0
w=.-~/9 o.y
h=.-~/11 o.y
xs=.({.y)+w*(i.%<:)1+x
ys=.h*(i:%j.)<.0.5+x*h%w
ys +/ xs
)

m=:|<:|z*^-.z=:1200 zl_clur _0.5 1.5j0.5
```

from fvj3

```
view_data m="1<./m
```



(assuming `~addons/media/image3/view_m.ijs` is loaded)




```
view_data m
```

POUTPOURRI:

Dragon Curve – Cantor Set

(Code archeology)

JoJ (j2 team)

journalofj at hotmail dot com

The basic structure is two line segments arranged on a 90 degree angle.

```
start=: 0 0, 1 0,: 1 1
```

And the iterative process involves a 90 degree rotation and a

```
step=: ],{: +"1 (2 2$0 _1 1 0) +/ .*~ |.@}: -"1 {:
```

Plotting

```
   load'plot'
NB. simple
    plot<"1|:step^:12 start
NB. colorful
    ([:pd[:<"1|:)every'reset';|.'show';step&.>^:(i.16)<start
```

```
NB. Cantor Set

require 'plot'

f=:3 : 0
X=.i.3^y
Y=.*./"1 ] 0 2 e.~ (y#3)#:X
pd 'reset'
pd 'type stick'
pd  X;Y
pd 'show'
)
```

Perimetric Complexity in fractal carpets

Perimetric complexity is a measure of the complexity of binary pictures. It is defined as the sum of inside and outside perimeters of the foreground, squared, divided by the foreground area, divided by 4π.

THE J GUIDEBOOK for Programming, Numerics and Graphics.

A international and interdisciplinary challenge !

Contact and info: journalofj at hotmail dot com

Collaborators in this issue:

Cliff Reiter	The Story of Fractals, Visualization and J
Roger Hui	The Zeros of the Partial Sums of e^z
Bo Jacoby	Probabilistic Prediction
David Alis	Elegant expressions

****

****

****

****

****

*

MPM Press **AN OPEN JOURNAL**

ISSN: 2174-9280 **Vol. 1, No.3, October 2012**

http://sites.google.com/site/jforscience/ journalofj@hotmail.com