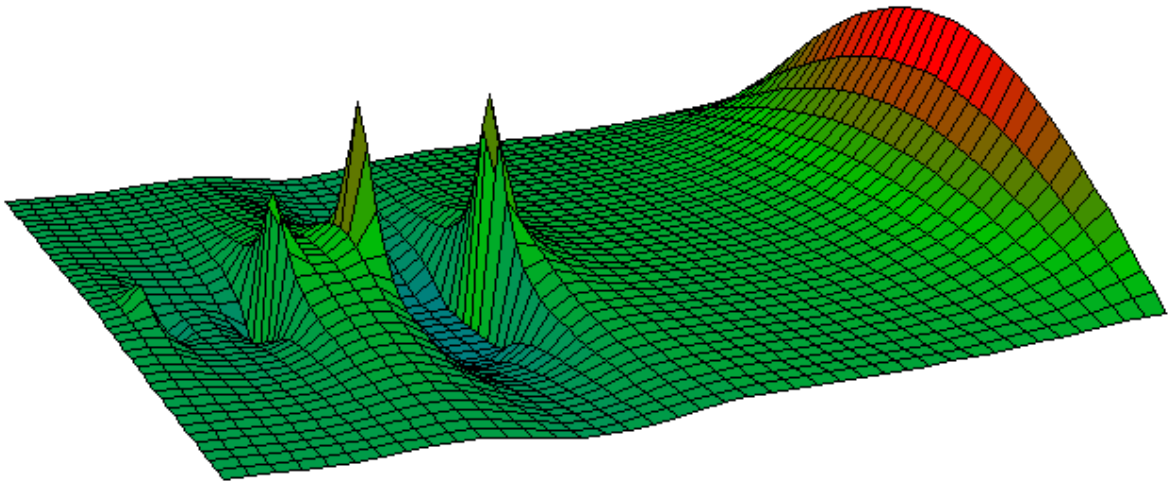
Journal of J

An interdisciplinary journal on J programming language and applications in science



Please Circulate

Journal of J. Focus Issue. December 2011

MPM press

ISSN: 2174-9280

An open access Journal

Vol.0, No.1 December 2011

Journal of J (J2-team)

J, a language for the science

Aims and scope

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

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

Journal of J aims to provide fast access to papers about science 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/

Table of contents

Editorial

Este número especial de fin de año pretendía centrase en aspectos particulares del lenguaje (raíces matemáticas del mismo) pero el tipo de contribuciones recibidas tienen que ver con aplicaciones del lenguaje.

El equipo de *Journal of J* sigue confiando en el proyecto y esperamos que esta revista sirva como punto de encuentro de todos los entusiastas en J. De igual manera, esperamos que las interesantes contribuciones de los programadores sirvan como lección y acicate para extender este lenguaje de programación.

Si el nivel de las contribuciones sigue aumentando esperamos continuar con este proyecto editorial en 2012.

Mikel Paternain (Editor)

mikelpater at Hotmail dot es

J-SYMBOL  an exercise suggested by the editor

Este número está dedicado a Orreaga, Ander y Katixa, constantes matemáticas en mi vida.

Author Notes:

- The principal reference(omitted by editorial mistake) for Digital Binary Sums ed)published in Journal Of J Vol.0, No.0 is *Digital Sums Problems by Andrew H. Osbaldestin in Fractals in the fundamental and applied Sciences, North –Holland, 1991.*

Letters to Editor

José Mario Quintana (e-mail in J forum and personal e-mail)

Some thoughts regarding the first issue

The dyadic Cantor function (#. #:) in the first article is related to the s(n) function `(+/"1 @ #:)` as follows,

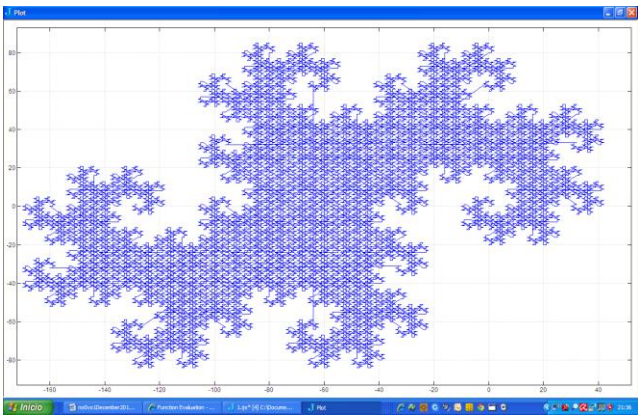
```
((+/"1 @ #:) -: (1&(#. #:)))
i.2^18
1
```

Self-similarities and other properties of the Cantor function family are apparent when they are graphed together:

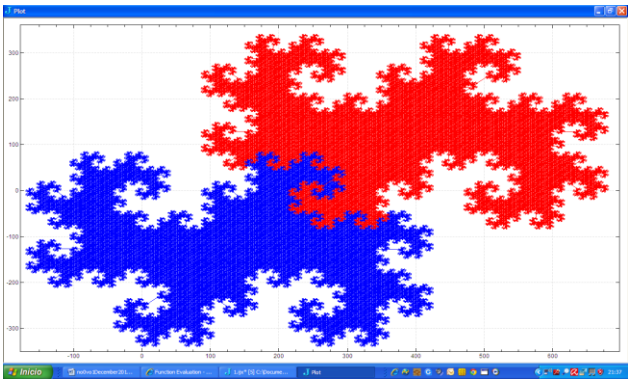
```
Cantor=. (#. #:) "0 _

NB. Setting a convenient rank

load'plot'

dl=. 0 # 6!:3 NB. delay

plot i.9 NB. Initial dummy
plot

plot _1.5 1.5 Cantor i.2^18
dl 7

plot _2 2 Cantor i.2^18
dl 3

plot _3 3 Cantor i.2^18
dl 3


plot 0.5j1.5 1.5j0.5 Cantor
i.2^18
dl 3
```

One of my favorite Cantor functions is 1j1&Cantor:

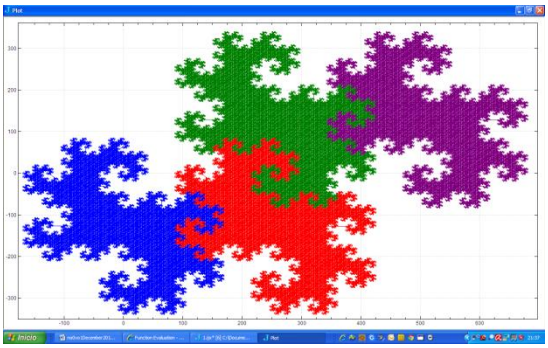
```
plot 1j1 Cantor i.2^0 18

NB. One dragon...
dl 3
```



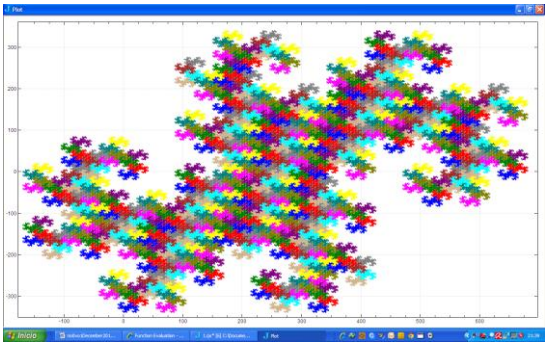
```
plot 1j1 Cantor i.2^1 17

NB. Made of 2 interlocking
dragons...
dl 3
```



```
plot 1j1 Cantor i.2^2 16

NB. Each made of 2
interlocking dragons...
dl 3
```



```
plot 1j1 Cantor i.2^9 9

NB. ... and so forth ...
dl 3
```



```
plot 1j1 Cantor i.2^17 1

 NB. One dragon made of 2^17
interlocking dragons
dl 3
```

I found this issue very stimulating. Thank you very much for sharing your efforts!

****

Raul Miller

That was interesting to read, thank you. I liked the bit about complex cantor functions, and the pictures are prodding at my desire to understand something that I have not fully characterized.

That said, there are a couple things that I would change, if I had the opportunity.

On page 6, you mention the need to simplify the results from q: -- in that context, I think that these results should have been mentioned:

```
_ q: !50x
47 22 12 8 4 3 2 2 2 1 1 1 1 1 1
__ q: !50x
2 3 5 7 11 13 17 19 23 29 31 37
41 43 47
47 22 12 8 4 3 2 2 2 1 1 1 1 1 1
```

And, at the bottom of page 12, J code was auto-corrected as if it were English.

(I should probably study this pdf further, but that will have to wait for time and opportunity.)

Raul

****

Bo Jacoby (2009-09-27)

Dear editors of the Journal of J.

Allow me to point your attention to my wikipedia contribution on

http://en.wikipedia.org/wiki/Bayesian_inference#Distribution_of_a_parameter_of_the_hypergeometric_distribution

and the J implementation of the formula on the discussion page

http://en.wikipedia.org/wiki/Talk:Bayesian_inference#Melcombe.27s_request.

Some wikipedia editors consider this to be original research on my part. I don't know. But it is a very useful and very elementary statistical formula, computing the mean value and the standard deviation of the number of elements of different types in a population, based on knowing the number of elements of different types in a sample.

These formulas

```
deduc=. (* (% +/)) ([ ,: %:@*)
*`%`:3@(-.@((,: , 1"_) %
+/@]))
  T =. -@(+ #)
```

```
induc =. (T&}: ,
}.)@(T~ deduc T)
```

are not found in the standard statistics books, as far as I know.

You may want to publish it in your journal.

Venlig hilsen,
Bo Jacoby.

****

Bo Jacoby (2011-11-27)

In the meantime I have replaced the program

```
   deduc =. (* (%
+/)) ([ ,: %:@*)
*`%`:3@(-.@((,: ,
1"_) % +/@]))
```

with the shorter, but equivalent, version

```
   deduc =.
*`%`:3"2@(,: (%:@* -
.))@((,: , 1:) %
+/@])
```

I made the change below as there is no reason to publish the longer version. If you want to include some examples, please let me know.

Venlig hilsen,
Bo.

****

Problems

Some questions suggested by the Editor and relationed with the journal articles. Also, questions and problems submitted by readers via e-mail.

Problem 0.1.1 *Vectorgrams*

In Clifford A. Pickover´s book *Computer Pattern chaos an beauty,* the *vectorgram* concept is used to studing DNA sequences.

First assing nucleotide input values as follows: G=1,C=1, A=0, T=0

Second: Create a DNA Vectorgram (2D figure) with the next algorithm

```
Variables:

(x,y)-current position in lattice

s-step size for walk on lattice

select(argument);

when('000')do x=x-s;y=y+s end;

when('001')do y=y+s end;

when('010')do x=x+s;y=y+s end;

when('011')do x=x-s end;

when('100')do x=x+s end;

when('101')do x=x-s;y=y-s end;

when('110')do y=y-s end;

when('111')do x=x+s;y=y-s end;

end;/*select*/

call Move Pen(x,y);
```

Write a script to convert a "DNA-Sequence" into a vectorgram. Send solution to

journalofj at hotmail dot com

Poutpourri *A section containing short articles on mathematical curiosities, algorithms, interesting equations, their graphics and J code.*

A interesting function

Matxin Lekim

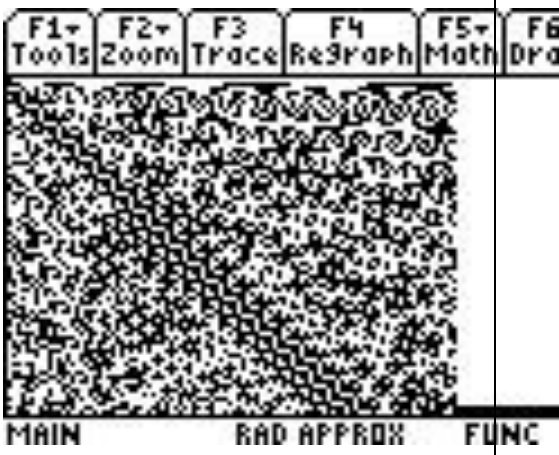
matxinlekim@hotmail.com

In the magazine *The journal of Chaos and Graphics,* (Clifford Pickover, Vol. 3, august 1988) appears a beautiful algorithm for reproducing interesting graphics. The code is by Rastislav Telgarsky

In my "old" **TI-89** , the code is a typical FOR-IF program. First define a function and second a program what use the function. I think that is self-explanatory:

```
@ Define a function
Tel(x,y)
Func( )
Int(x*√(x*y))
EndFunc

@ Define the program
Prgm
ClrDraw
For x,1,100,1
For y,1,100,1
Mod(Tel(x,y),2)->z
If z=1 Then
PxlOn(x,y) @Draw the point (x,Y)
Else
PxlOf(x,y)
EndFor
EndFor
EndPrgm
```

The next figure shows the result with the TI-89 resolution



Asking in J-forum I found several answers for Tel(x,y) function..but how to translate to J the previous code?

****

Mathematics for Radio and Electronics using the J programming language

Alan Holt

December 2, 2011

1 Introduction

Mathematics is the foundation of of radio and electronic engineering. Computers, with their high-resolution graphical capabilities, are useful for visualising mathematical expressions. Spreadsheet applications, such as Microsoft Excel, possess some mathematical functionality but is somewhat unwieldy for engineering applications. There are a number of excellent specialist computer algebra systems, such as Maple and Mathematica. While these systems are ideal for engineering applications, they can be prohibitively expensive.

In this article, we introduce the J programming language and demonstrate, through worked examples, how it can be used for radio and electronic engineering. J is rich in mathematical functionality enabling the user to build mathematical expressions efficiently on the command-line.

Note that, all the examples in this paper were performed using version 602 of the J programming language.

2 Some Circuit Analysis

Consider a simple example of calculating combined resistance R_S of resistors in

series:

$$R_S = R_1 + R_2 + \ldots R_n \quad (1)$$

Consider three resistors in series, $R_1 = 200\Omega, R_2 = 30k\Omega \text{ and } R_3 = 1M\Omega$. The total resistance can be calculated by simply adding the three resistor values:

```
+/ 200 + 30e3 + 1e6
1030200
```

Not too many surprises here but note the use of scientific notation. Instead of writing

30000 and 1000000 (Ω), we use e3 and e6 as a convenient means of expressing k Ω and M Ω, respectively. J is not limited to scalar quantities, it can also deal with higher dimension objects such as lists, arrays and matrices (of any dimension), usually without any extra programming. Another approach to calculating resistors in serial is to use the summation +/ function:

```
RS =: +/ 200 30e3 1e6

RS

1030200
```

Here, the resistor values are passed as list to the function. The total resistance R_p of two resistors in parallel is:

$$R_p = \frac{R_1 R_2}{R_1 + R_2} \quad (2)$$

The formula above is demonstrated with dyadic J expression below (using resistor values R1 = 20;R2 = 30 Ω):

```
20 (*%+) 30

12
```

The expression *%+ forms a construct called a fork. The * and + operators return the product and sum of 20 and 30 () respectively. The result of the product is then divided by the result of the sum with the % operator. The expression above is somewhat limited as it only works for two resistors and is merely a special case of the formula for calculating an arbitrary number of parallel resistors:

$$\frac{1}{R_P} = \frac{1}{R_1} + \frac{1}{R_2} \ldots \frac{1}{R_n} \quad (3)$$

One approach in J is to represent the reciprocal of the resistor values as rationales, so 1/20 is expressed as `1r20` in J. Here we sum the reciprocals (with +/) to find $1/R_p$:

```
+/ 1r20 1r30 1r30 1r100

19r150
```

We apply the reciprocal operator % to the sum operator to calculate R_p:

```
] RP =: % @: (+/) 1r20 1r30 1r30 1r100

150r19
```

The at conjunction `@:` is a sequencing construct and controls how verbs are combined

and applied to arguments. (note the right operator] is used to display the result of the assignment).

If we want to avoid using rationales, then we can use `%` to calculate reciprocals of the resistance values before to summing them:

```
] RP =: % @: (+/ @: %) 20 30 30 100

7.89474
```

With this expression, we are not limited to calculating resistors in parallel. Given J's tacit programming style we can also use it for calculating inductors in parallel. Now we look at examples of reactance (inductive and capacitative). For this topic, the power of J becomes apparent because of its support for complex numbers.

Inductive and capacitative reactance is given by the two equations below, respectively:

$$X_L = j\omega L \text{ and } X_C = \frac{1}{j\omega C} \quad (4)$$

Where $\omega = 2\pi f$. There is some amount of commonality between the the two equations. We can use this to help us write `lreact` and `creact`, which will be the two functions for calculating inductive and capacitive reactance respectively.

We define a *multiply-by-two-pi* function (`twopi`) and a function `g`:

```
twopi =: 2p1&*       NB. multiply by 2PI

g =: j. @ (twopi @: *)
```

The function `g` is just and intermediate function which we use for convenience to write `lreact` and `creact`. The function `lreact` is merely an alias of `g` and `creact` is its reciprocal:

```
lreact =: g

creact =: % @: g
```

Consider an inductor $L = 25\mu F$ and a set of frequencies $f = 10Hz, 1kHz, 1MHz \text{ and } 1GHz$,

we calculate the reactance:

```
L =: 25e_6              NB. inductance 25 mu Henries

f =: 100 1e3 1e6 1e9    NB. frequencies 100 to 1 GHz
```

Find the inductive reactance in complex form:

```
] XL =: L lreact f

0j0.015708 0j0.15708 0j157.08 0j157080
```

The variable `XL` stores the result in complex form but we use `*.` to display in

magnitude and phase form:

```
*. XL

0.015708 1.5708

0.15708 1.5708

157.08 1.5708
```

```
157080 1.5708
```

It can be seen that as the frequency increases, the reactance increases (as we would expect). Furthermore, the result shows the phase is 90º (1.5708 radians).

Calculate the reactanace for a 100nF capacitor.

```
C =: 100e_9       NB. 100 nano Farads

*. XC =: C creact f

15915.5 _1.5708

1591.55 _1.5708

1.59155 _1.5708

0.00159155 _1.5708
```

We can see that the magnitude of reactance is decreasing with increasing frequency

(again, as we would expect) and the phase is -90º (-1.5708 radians).

The resonant frequency f_0 of an LC circuit is given by:

$$f_0 = \frac{1}{2\pi\sqrt{LC}} \qquad (5)$$

The corresponding J function `resf` is defined, thus:

```
resf =: %@: (twopi @: %: @: *)
```

It is a dyadic function taking inductance and capacitance as left and right arguments spectively. The function is actually commutative, so it does not actually matter which arguments are inductance or capacitance:

```
]f0 =: 25e_6 resf 100e_9

100658
```

The result shows that the resonant frequency, $f_0 \approx 100Hz$. We verify this with

`lreact` and `creact`:

```
XL =: 25e_6 lreact f0

XC =: 100e_9 creact f0

*. XL,XC

15.8114 1.5708

15.8114 _1.5708
```

Assuming 5V is dropped across the inductor, we can compute the current (using

Ohms law):

```
*. IL =: 5 % XL

0.316228 _1.5708
```

The current is 0.32 A, and the negative phase indicates that current lags the voltageby 90º Similarly, for the capacitor, the voltage lags the current by 90º:

```
*. IC =: 5 % XC

0.316228 1.5708
```

Load the trigonometry and plot package:

```
load 'trig'

load 'plot'
```

Define functions for the current in the inductor and capacitor:

```
il =: 0.32 &* @: sin @: (_1.5708 &+ )

ic =: 0.32 &* @: sin @: (1.5708 &+ )
```

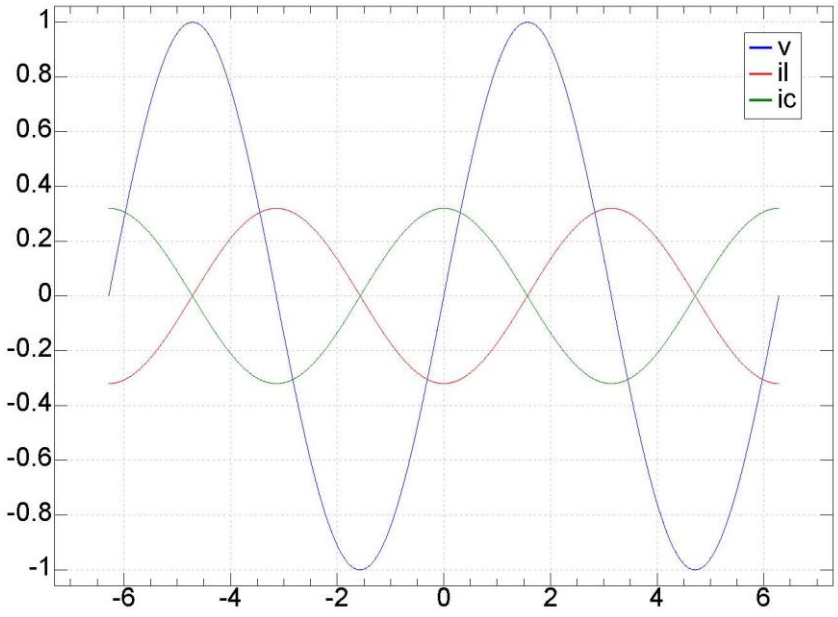


Figure 1: Current and voltage in an LC Circuit

Define a function for the voltage:

```
v =: sin
```

Here we scale the voltage by a fifth (to unit amplitude) purely for graphing:

```
opts =: 'keypos right;key v il ic;keystyle for'
```

```
opts plot _2p1 2p1 ; 'v`il`ic'
```

The plot command produces the graph in Figure 1. We can see that the current through the inductor (red line) lags the voltage (blue line). However, the voltage lags the current through the capacitor (green line).

3 Logarithms and Decibels

Logarithms are used extensively in radio and electronics. To perform $log_{10}$with:

```
10 ^. 1 10 100 1000
```

```
0 1 2 3
```

For convenience, we can create a *log-to-the-base-ten* function with the help of the *bond* `&` conjunction:

```
log10 =: 10&^.
```

```
log10 1 10 100 1000
```

```
0 1 2 3
```

We can take this example further and create a decibel function ($10log_{10}$):

```
decibel =: 10&* @: log10
```

```
decibel 1 2 4 8 16 32
```

```
0 3.0103 6.0206 9.0309 12.0412 15.0515
```

We will complete this section by writing a function to calculate free-space loss:

$$L_{fspl} = 20log_{10}f + 20log_{10}d + K \qquad (6)$$

where f is the frequency, d is the distance and K *is* a constant. The *constant* K is used to control the units of frequency and distance. The function `fspl` is defined,

thus:

```
fspl =: + +/ @: (20&* @: log10)
```

The function `fspl` accepts frequency f and distance d as right arguments (though they can be supplied in any order). The constant `K` is supplied as the left argument. If we use $K = -147.55$ then f is in hertz and d in meters. For example, the freespace loss over a distance of 1,000 meters in the 10 GHz range is:

```
_147.55 fspl 10e9 1e3 NB. Hz/meters
```

```
112.45
```

A value of K = 92:45 allows us to express frequency and distance in GHz and kilometres respectively:

```
92.45 fspl 10 1 NB. 10 GHz, 1 kilometre

112.45
```

If (say) we need to calculate free-space loss using units of MHz and miles, then we can create a new function by bonding the constant to the `fspl` function. In this case K = 36:58:

```
fsloss =: 36.585&fspl
```

Now we can calculate free-space loss without the constant:

```
fsloss 10e3 0.6214 NB. 0.6214 * mile = 1km

112.452
```

4 Filters

For our final topic we examine filters. Consider the filter transform:

$$H_1(s) = \frac{s}{s^2+s+1} \qquad (7)$$

This is implemented in J with:

```
H1 =: % (>: @ +*:)
```

The *squared* function (`*:`), forms a hook with the plus function (+) resulting in $s^2 + s$. The J expression `>: @ +*:` yields $s^2 + s + 1$ and forms a hook with `%` to give $s/(s^2 + s + 1)$. Generate a matrix of values on the complex plane:

```
]s =: j./ ~ (i:2j5)

_2j_2 _2j_1.2 _2j_0.4 _2j0.4 _2j1.2 _2j2

_1.2j_2 _1.2j_1.2 _1.2j_0.4 _1.2j0.4 _1.2j1.2 _1.2j2

_0.4j_2 _0.4j_1.2 _0.4j_0.4 _0.4j0.4 _0.4j1.2 _0.4j2

0.4j_2 0.4j_1.2 0.4j_0.4 0.4j0.4 0.4j1.2 0.4j2

1.2j_2 1.2j_1.2 1.2j_0.4 1.2j0.4 1.2j1.2 1.2j2

2j_2 2j_1.2 2j_0.4 2j0.4 2j1.2 2j2
```

Apply the transform (and compute the magnitude):

```
| H1 s

0.464991 0.594469 0.661541 0.661541 0.594469 0.464991
```

```
0.593237 1.00307 1.03975 1.03975 1.00307 0.593237

0.624765 1.75412 0.934539 0.934539 1.75412 0.624765

0.468986 0.584705 0.359327 0.359327 0.584705 0.468986

0.342518 0.366112 0.338546 0.338546 0.366112 0.342518

0.270914 0.28513 0.286204 0.286204 0.28513 0.270914
```

We recalculate s for 50 intervals:

```
s =: j./ ~ (i:2j50)
```

For brevity, we omit displaying the result to the screen (because it is a 50 x 50 array) and plot the results instead. The example below produces the 3D surface graph in Figure 2 over s:

```
opts =: 'surface;noaxes;boxed'

opts plot | H1 s
```

Now consider the transform expression:

$$H_2(s) = \frac{1}{s^3+2s^2+2s+1} \tag{8}$$

We could implement this in J with:

```
H2 =: % @ ((**:) + >:@(+:@+ *:))

H2 _2 j./ (i:2j2)

0.0702703j0.0216216 _0.333333 0.0702703j_0.0216216
```

However, the expression above is somewhat unwieldy. For even higher order polynomials, this approach would yield overly complex expressions. The function `p.` provides a more convenient method of expressing polynomials. The denominator in H_2 is a polynomial with coefficients 1,2,2 and 1. We can evaluate the polynomial $s^3 + 2s^2 + 2s + 1$ thus:

```
1 2 2 1 p. _2 j./ i:2j2

13j_4 _3 13j4
```

We can see that the coefficients are passed as left arguments and variables as right arguments. Evaluating $H_2$is now straight forward:

```
% 1 2 2 1 p. _2 j./ i:2j2

0.0702703j0.0216216 _0.333333 0.0702703j_0.0216216
```

H_2 can be redefined as a function (using `p.`):

```
H2 =: % @: (1 2 2 1 & p.)
```

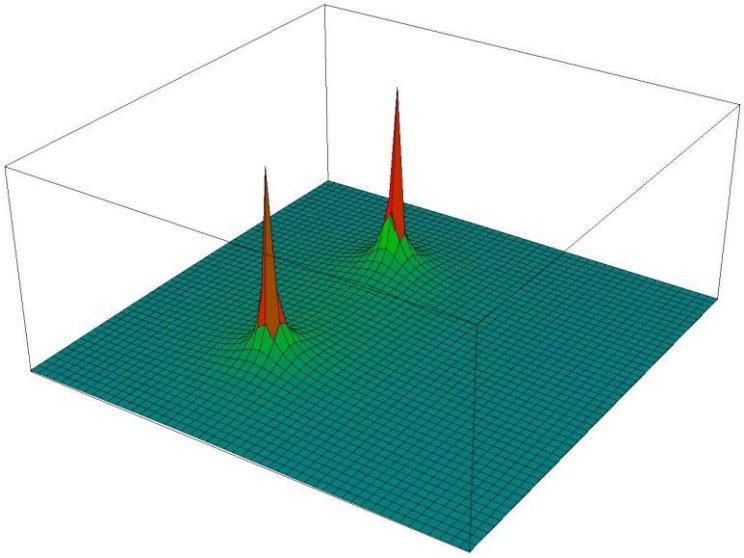


Figure 2: Transform of $H_1 = s/s^2 + s + 1)$

The conjunction `@:` is used to prevent `%` forming a hook with the polynomial denominator and ensures the reciprocal function is applied to polynomial. The graph in Fig 3 shows a contour plot which reveals H_2 has three poles:

```
opts =: 'surface;noaxes;grid 1 1'

opts plot | H2 s
```

5 Summary

J is a powerful analytical tool which can be used in many disciplines but in this article we demonstrated, through examples, how it can be applied to RF and electronics engineering. The strengths of the J programming language are summerised below:

- Rich in mathematical functionality.
- Powerful (albeit unconventional) programming style.
- Support for many data type, for example, rationales and complex numbers.
- Powerful graph ploting features for visualising expressions.

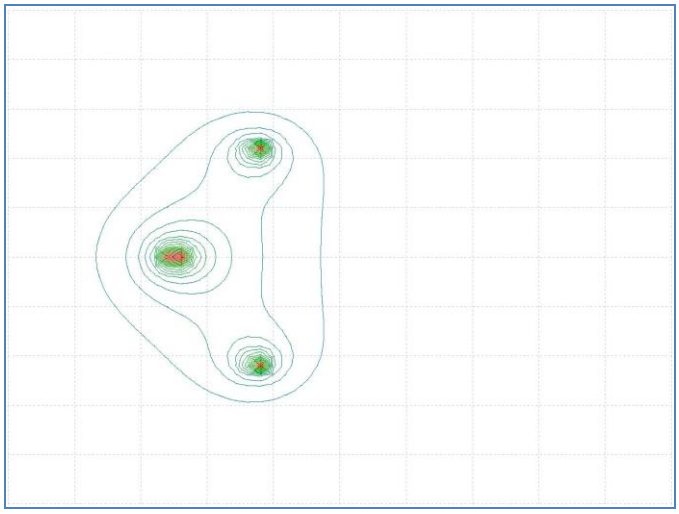


Figure 3: Transform of $H_2(s) = 1/(s^3 + 2s^2 + 2s + 1)$

Furthermore, J is free to download and use. It is also available on many platforms, making it highly portable.

References

[1] Alan Holt. Network Performance Analysis using the J Programming Language. Springer, 2007.

[2] Norman Thomson. J for engineers and computing professionals. IEE Computing & Control Engineering Journal, 12(5):212–216, 10 2001.

[3] Norman Thomson. J: The Natural Language for Analytical Computing. Research Studies Press Ltd, Baldock Hertfordshire, England, 2001.

Correlation in Symbolic Sequences

J in bioinformatics...

Mikel Paternain

mikelpater@hotmail.es

1.Introduction

DNA sequences are symbolic sequences, and the possible symbols are *A*, *C*, *G*, and *T*, representing the four nucleotide bases of a DNA strand — adenine, cytosine, guanine, thymine.

 an interesting exercise would be to design a script in J able to read a FASTA format file. **Send your solution to** `journalofj at hotmail dot com`

Next is a typical FASTA sequence of symbols with biological significance (dogma of molecular biology):

```
>gi|5524211|gb|AAD44166.1| cytochrome b [Elephas maximus maximus]
LCLYTHIGRNIYYGSYLYSETWNTGIMLLLITMATAFMGYVLPWGQMSFWGATVITNLFSAIPYIGTNLV
EWIWGGFSVDKATLNRFFAFHFILPFTMVALAGVHLTFLHETGSNNPLGLTSDSDKIPFHPYYTIKDFLG
LLILILLLLLLALLSPDMLGDPDNHMPADPLNTPLHIKPEWYFLFAYAILRSVPNKLGGVLALFLSIVIL
GLMPFLHTSKHRSMMLRPLSQALFWTLTMDLLTLTWIGSQPVEYPYTIIGQMASILYFSIILAFLPIAGX
IENY
```

Many interesting operations in *bioinformatics* are straightforward and simple in J. For example: count symbols in a sequence, extract symbols and more:

```
SEQ_DNA=. 'LCLYTHIGRNIYYGSYLYSETWNTGIMLLLITMATAFMGYVLPWGQMSFWGATVIT'

# SEQ_DNA
56

~. SEQ_DNA
LCYTHIGRNSEWMAFVPQ
```

In this context we think that J is a powerful programming language for bioinformatics !

Richard F. Voss [1] proposed a method for calculating *spectral density* an *autocorrelation* for non-numeric sequences, such as DNA sequences.

The standard definition of correlation requires a definition of *equal-symbol multiplication*:

(1)

Let be `Xn` and `Xm` two sequences of DNA. For example:

```
Xn=.'ACGATGCAATGCCGTAGTAATGGCAGTCATTAGCACCTGATGGCTAGTCTTCAAT'
```

```
Xm=.'ACGATGCAATGCCGTAGCAATGGCAGTCATTAGCATCTGATGGCTAGTCTTCAAT'
```

The implementation of (1) is very easy, trivial, in J,

```
Xn = Xm
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

#Xn
55
```

In J, the treatment of logical values as numbers is useful to establish a first comparation between sequences

```
+/ Xn = Xm
53
```

This value is very close to `#Xn` and `#Xm` indicating similarity between sequences.

2.Binary Indicators and Correlation Function

In [1], Voss also defined the ***binary indicator function*** U_k for each of the k=1,2,3…K allowed symbols. The U_k select the element of X_n that are equal to symbol k

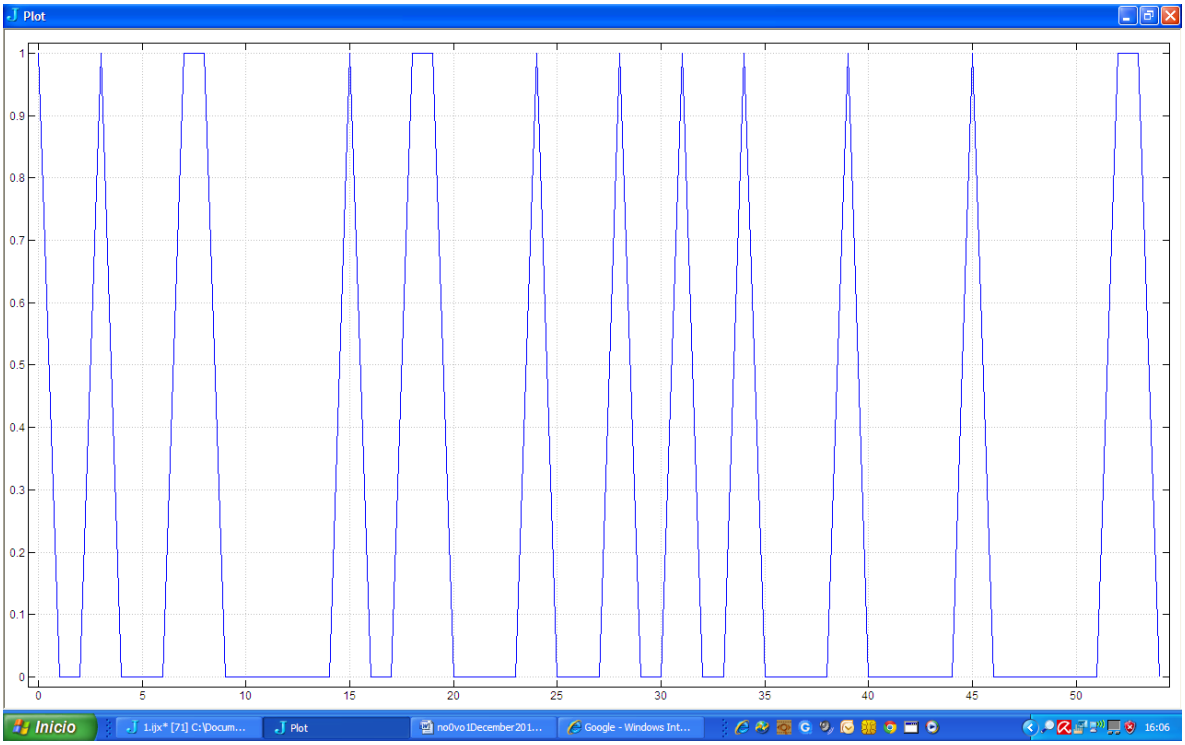
$$U_k[X_n] = \begin{cases} 1 \; if \; X_n = k \\ 0 \; otherwise \end{cases} \qquad (2)$$

In our study case (DNA sequences with four nucleotide bases)the symbols are A,C,G and T. A possible verb to implement (2) in J is:

```
]UA=. 'A'&=   NB. Select A symbol of sequences
]UC=. 'C'&=   NB. Idem with C
]UT=. 'T'&=   NB. With T
]UG=. 'G'&=   NB. And select G

UG Xn
0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 0 1 0 0 0 0 1 1 0 0 1 0 0 0 0 0 0 1
0 0 0 0 0 1 0 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0
   UG Xm
0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 0 1 0 0 0 0 1 1 0 0 1 0 0 0 0 0 0 1
0 0 0 0 0 1 0 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0
```

In bioinformatic suites, a common tool is visualising nucleotide bases (A,T, G or C) in a sequence. This is very easy with J:

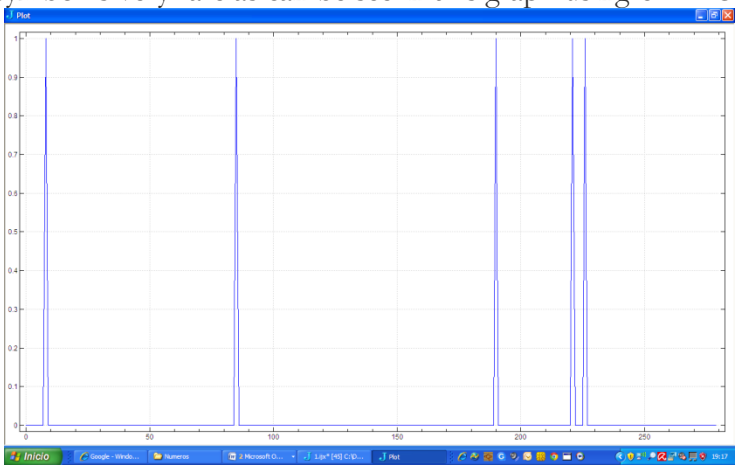
```
load 'plot'
plot UA Xn
```



For large sequences as FASTA example in this paper, we can make sub-sequences and next append with `,` verb:

```
a=.'LCLYTHIGRNIYYGSYLYSETWNTGIMLLLITMATAFMGYVLPWGQMSFWGATVITNLFSAIPYIGTNL
V'
b=.'EWIWGGFSVDKATLNRFFAFHFILPFTMVALAGVHLTFLHETGSNNPLGLTSDSDKIPFHPYYTIKDFL
G'
c=.'LLILILLLLLLALLSPDMLGDPDNHMPADPLNTPLHIKPEWYFLFAYAILRSVPNKLGGVLALFLSIVI
L'
d=.'GLMPFLHTSKHRSMMLRPLSQALFWTLTMDLLTLTWIGSQPVEYPYTIIGQMASILYFSIILAFLPIAG
X'
```

In this long sequence, the R symbol is very rare as can be see in this graph using `UR verb`:



```
plot UR a,b,c,d
```

Others verbs can be use to obtain significant dates. For example:

```
# a,b,c,d
280

+/UA a,b,c,d
17

(+/UA)%# a,b,c,d
```

The frequency of a concrete symbol (important measure) is easy to compute:

```
(+/UA)%# a,b,c,d
0.00357143
```

Following [1] the *equal-symbol autocorrelation function* for a individual symbol k is

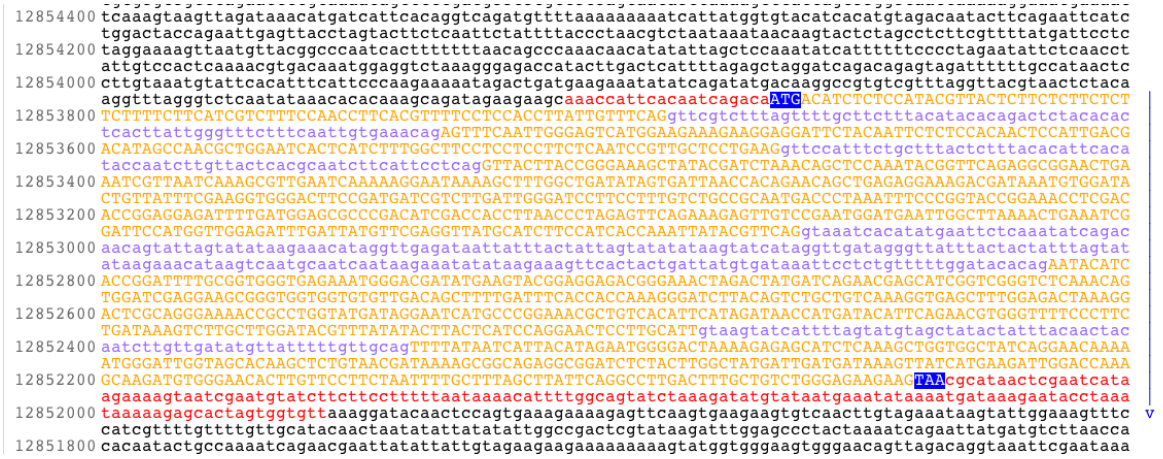
$$C(\tau) = \sum_{i=1}^{N} \sum_{k=1}^{N} U_k[X_n] U_k[X_{n+\tau}] \qquad (3)$$

Is a exersice for the reader implement a J-verb for equal-symbol autocorrelation function (3).
Send your solution to journalofj@hotmail.com

References:

[1] 1/f Noise and Fractals in DNA-base sequences, Richard F. Voss in *Applications of Fractals and Chaos,* Springer-Verlag, 1993.

[2]Another interesting use of DNA Sequence is in *Fractals and Visualization for the J User Conference 2000.*Clifford A. Reiter.



See Problems Section for another interesting question about J and bioinformatics.

****

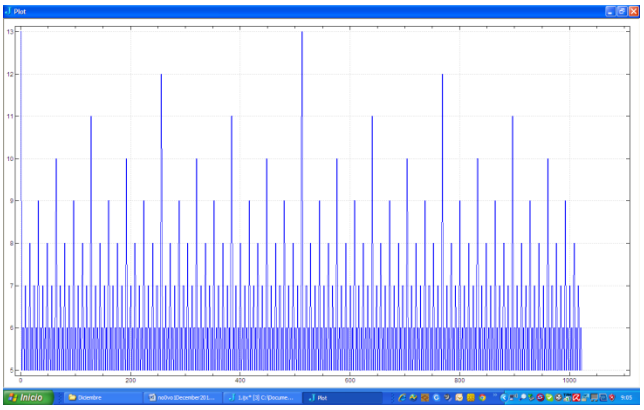
Short Note: Fractal sequences

J2 Team

Define a verb `mm=:, , . ,`

```
load 'plot'

plot +/ mm^:10.~1,,1
```

The result is a fractal number sequence.



J, a language for the science

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

An open access Journal

Call of papers

Authors are invited to submit papers for publication. Manuscripts should be submitted by

e-mail to journalofj@hotmail.com

Periodicity
Six issues per year

Note: Journal of J is an open and free Journal. If you send a paper to Journal of J, anyone is free to copy, distribute, use,change, make derivative works (based in your work)and display your work published in Journal of J

Guidelines for Contributors (see Vol.0,No.0)

Papers must be sent in word format or plain text. We recommend, for example, a style like the APL Quote Quad Article Template (see http://sigapl.org/qq.htm#submit)

For others formats contact with editor journalofj@hotmail.com

There is no strict limit on the length of a paper

****

Anuncio

El equipo de Journal of J ha puesto a punto un manual con ejemplos básicos del lenguaje Q. Las personas interesadas pueden solicitarlo directamente a journalofj @ Hotmail.com. Asunto: *Q con ejemplos*.

Visit thttp://sites.google.com/site/jforscience/

Collaborators in this issue:

J2-Team	Edition and several notes, questions, problems, new ideas, etc.
Alan Holt	Paper: Mathematics for Radio and Electronics.
Matxin Lekim	A interesting function in Poutpourri section.
Mikel Paternain	Correlation in symbolic sequences.
José Mario Quintana	Dragon curves in Letters to Editor.
Raul Miller	Questions about No.0,Vol.0 in Letters to Editor.
Bo Jacoby	Statistic formulas in Letters to Editor.

****

Acknowledgements

Journal of J is a non professional journal with a main aim: extend the knowledge of J programming language.

Special thanks to collaborators in this issue.

MPM press

An open access Journal

ISSN: 2174-9280

Vol.0, No.1

December 2011

Journalofj at hotmail dot com