MoHPC The Museum of HP Calculators

Search Member List Calendar Help

**Welcome back, Valentin Albillo**. You last visited: Yesterday, 10:28 PM **Current time:** 05-01-2019, 01:50 AM (**User CP** — Log Out)
View New Posts | View Today's Posts | Private Messages (Unread 0, Total 145) Open Buddy List

**HP Forums / HP Calculators (and very old HP Computers) / General Forum / [VA] SRC#003- New Year 2019 Special**

**Pages (2):** 1 2 Next »

NEW REPLY

## [VA] SRC#003- New Year 2019 Special

01-16-2019, 12:42 AM **Post: #1**

**Valentin Albillo**
Senior Member

Posts: 347
Joined: Feb 2015
Warning Level: 0%

**[VA] SRC#003- New Year 2019 Special**

.
**Hi, all:**

Welcome to my brand new ***SRC#003***, this time commemorating the *New Year **2019***.

There will be some (hopefully) interesting things discussed here but first of all there's a simple assignment for you to tackle with your preferred **HP** calculator *(not Excel, Matlab, Mathematica, Python, Java, Haskell, Wolfram Alpha, etc., there are other forums/threads for that)*, namely:

**Write a program which implements the following procedure:**

**1)** Create this 3x3 matrix **M**:

$$\mathbf{M} = \begin{vmatrix} \pi & 2019 & 2019 \\ 1 & \pi & 2019 \\ 1 & 1 & \pi \end{vmatrix}$$

**2)** Compute the successive powers of **M** (i.e: $\mathbf{M}^2$, $\mathbf{M}^3$, etc.) and for each power **n** compute the value of $\mathbf{M}^n(1,3)$ / $\mathbf{M}^n(2,3)$ until it converges to some limit.

***What is the numerical value of this limit ? What do you think this procedure is actually computing ?***

You can use any **HP** calculator of your choice (**M**inimum **R**ecommended **M**odel would be the **HP-11C** or better) but, again, write your code only in **RPN**, **RPL** or **71-BASIC**, please, and it would be better if you don't peruse the Internet, just your programming skills and math intuition.

In a few days I'll give a *5-line*, *144-byte* program for the **HP-71B** (easiest code to understand), plus extensive comments and further discussion.

Meanwhile, let's see what you come up with. 🙂

V.

.

PM FIND EDIT QUOTE REPORT

01-16-2019, 05:51 AM **Post: #2**

**Paul Dale**
Senior Member

Posts: 1,455
Joined: Dec 2013

**RE: [VA] SRC#003- New Year 2019 Special**

I gain insight.

Pauli

PM FIND QUOTE REPORT

01-16-2019, 05:57 AM (This post was last modified: 01-16-2019 05:58 AM by Carsen.) **Post: #3**

**Carsen**
Member

Posts: 163
Joined: Jan 2017

**RE: [VA] SRC#003- New Year 2019 Special**

Hello all,

After just silently reading these challenges for awhile, I have decided that I'll take a stab at this problem using UserRPL on my 50g. First, I store the array in a global variable named 'M'. Then I execute the the following code.

**Code:**

```
SPOILERS!!!
```

It took my 50g 239.15 seconds to complete, getting a result of 12.6389823195. I believe that this is correct.

However, I don't have an answer to the more important question of "What do you think this procedure is actually computing?"

PM FIND QUOTE REPORT

01-16-2019, 06:24 AM (This post was last modified: 01-16-2019 06:37 AM by Thomas Okken.) **Post: #4**

**Thomas Okken**
Senior Member

Posts: 763
Joined: Feb 2014

**RE: [VA] SRC#003- New Year 2019 Special**

12.638982319390552860486073878715 31

**Code:**

```
SPOILERS!!!
```

EMAIL PM FIND QUOTE REPORT

01-16-2019, 11:33 AM (This post was last modified: 01-16-2019 11:36 AM by Thomas Klemm.) **Post: #5**

**Thomas Klemm**
Senior Member

Posts: 1,449
Joined: Dec 2013

**RE: [VA] SRC#003- New Year 2019 Special**

The vector

$$e = \begin{bmatrix} u^2 \\ u \\ 1 \end{bmatrix}$$

is eigenvector to the matrix

$$M = \begin{bmatrix} a & u^3 & u^3 \\ 1 & a & u^3 \\ 1 & 1 & a \end{bmatrix}$$

with eigenvalue $\lambda = u^2 + u + a$ since

$$\begin{bmatrix} a & u^3 & u^3 \\ 1 & a & u^3 \\ 1 & 1 & a \end{bmatrix} \begin{bmatrix} u^2 \\ u \\ 1 \end{bmatrix} = \begin{bmatrix} au^2 + u^4 + u^3 \\ u^2 + au + u^3 \\ u^2 + u + a \end{bmatrix} = (u^2 + u + a) \begin{bmatrix} u^2 \\ u \\ 1 \end{bmatrix}$$

For the given values $a = \pi$ and $u^3 = 2019$ it appears that $\lambda$ is the biggest eigenvalue.
Thus the vector $M^n v$ will eventually converge to a multiple of the eigenvector $e$ for every vector $v \neq 0$.
Since we look only at values in the last column of $M^n$ we can just as well calculate:

$$M^n \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Thus the ratio

$$\frac{M^n_{1,3}}{M^n_{2,3}}$$

converges to the value

$$\frac{e_1}{e_2} = \frac{u^2}{u} = u$$

For the given value $u^3 = 2019$ this means $u = \sqrt[3]{2019} \doteq 12.63898232$.

Here's a program for the **HP-48GX**:

**Code:**

```
«
  @ (n M - ratio )
  [ 0 0 1 ] ROT
  1 SWAP START
    OVER SWAP *
  NEXT
  SWAP DROP
  DUP 1 GET
  SWAP 2 GET /
»
```

## Example:

With the given matrix **M** in a variable:

```
200 M
→
12.6389823194
```

Or then using what we know from above:

```
M EGV DROP { 2 1 } GET INV
(12.6389823194,0)
```

This works since $e_1$ of eigenvectors appears to be 1.

Cheers
Thomas

PM FIND QUOTE REPORT

01-16-2019, 11:50 AM **Post: #6**

**Thomas Klemm**
Senior Member

Posts: 1,449
Joined: Dec 2013

**RE: [VA] SRC#003- New Year 2019 Special**

> **Thomas Okken Wrote:** (01-16-2019 06:24 AM)
>
> **Code:**
>
> ```
> SPOILERS!!!
> ```

A similar calculation can be done:

$$\begin{bmatrix} a & u^4 & u^4 & u^4 \\ 1 & a & u^4 & u^4 \\ 1 & 1 & a & u^4 \\ 1 & 1 & 1 & a \end{bmatrix} \begin{bmatrix} u^3 \\ u^2 \\ u \\ 1 \end{bmatrix} = \begin{bmatrix} au^3 + u^6 + u^5 + u^4 \\ u^3 + au^2 + u^5 + u^4 \\ u^3 + u^2 + au + u^4 \\ u^3 + u^2 + u + a \end{bmatrix} = (u^3 + u^2 + u + a) \begin{bmatrix} u^3 \\ u^2 \\ u \\ 1 \end{bmatrix}$$

Cheers
Thomas

PM FIND QUOTE REPORT

01-17-2019, 08:04 PM **Post: #7**

**DavidM**
Senior Member

Posts: 722
Joined: Dec 2013

**RE: [VA] SRC#003- New Year 2019 Special**

Not being well-versed in the mathematics behind this, my attempt was simply a naïve/brute force loop to compare successive results until the first duplicate was encountered (50g/RPL):

**Code:**

Result: 12.6389823194
Execution time: about 12s on a real 50g.
Size: 146.5 bytes, of which 84 bytes is spent building the initial matrix. If I take that part out and require the initial matrix being left on the stack as an argument, the size drops to 62.5 bytes.

A similar approach using Thomas' first optimization:

**Code:**

Result: 12.6389823194
Execution time: about 8.5s on a real 50g
Size: 149 bytes (with embedded matrix build), 65 bytes (initial matrix as argument)

All of which pales in comparison, of course, to Thomas' ultimate eigenvalue solution. If I assume the initial matrix is on the stack as an argument, the following code gives the result in the same format as the other routines (I optimized the retrieval of the needed matrix element as well as adding "RE" to standardize the result):

**Code:**

```
\<<
   EGV DROP 4. GET INV RE
\>>
```

Result: 12.6389823194
Execution time: about 3.8s on a real 50g
Size: 28 bytes

Great job Thomas! (as usual). And thanks to Valentin for an interesting problem -- I hope to see more about the underlying concepts. I think this is the first time I've actually encountered eigenvalues being used outside of a Linear Algebra course many moons ago. I've forgotten most everything I was supposed to have learned in that class, I'm afraid.

PM FIND QUOTE REPORT

01-18-2019, 07:04 AM **Post: #8**

**Thomas Klemm**
Senior Member

Posts: 1,449
Joined: Dec 2013

**RE: [VA] SRC#003- New Year 2019 Special**

> **Valentin Albillo Wrote:** (01-16-2019 12:42 AM)
>
> **M**inimum **R**ecommended **M**odel would be the **HP-11C** or better

This program is for the **HP-11C**:

**Code:**

```
001-    42 34    CLEAR REG
002-    44 25    STO I
003-       33    R↓
004-     44 0    STO 0
005-        1    1
006-     44 3    STO 3
007-42,21, 0    ▸LBL 0
008-     45 3    RCL 3
009-     45 0    RCL 0
010-       20    ×
011-    42 16    π
```

## Example:

2019
ENTER
200
R/S

(this may take some time ...)

```
12.63898232
```

## Registers:

```
0: 2019 (or whatever you entered)
1: x
2: y
3: z
```

Starting with:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

The following matrix multiplication is iterated:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \to \begin{bmatrix} \pi & 2019 & 2019 \\ 1 & \pi & 2019 \\ 1 & 1 & \pi \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

The resulting vector is reduced in lines 29-31 to avoid overflow.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \to \begin{bmatrix} 1 \\ \frac{y}{x} \\ \frac{z}{x} \end{bmatrix}$$

Cheers
Thomas

PM FIND QUOTE REPORT

01-18-2019, 07:44 PM **Post: #9**

**Thomas Klemm**
Senior Member

Posts: 1,449
Joined: Dec 2013

**RE: [VA] SRC#003- New Year 2019 Special**

The translation for the **HP-67** is straightforward:

**Code:**

```
001: 31 43     :    CL REG
002: 35 33     :    ST I
003: 35 53     :    R↓
004: 33 00     :    STO 0
005: 01        :    1
006: 33 03     :    STO 3
007: 31 25 00  :   ▸LBL 0
008: 34 03     :    RCL 3
009: 34 00     :    RCL 0
010: 71        :    ×
011: 35 73     :    π
```

## Example:

```
2019
ENTER↑
200
R/S
```

(lots of blinkenlichten ...)

```
12.63898232
```

Cheers
Thomas

PM FIND QUOTE REPORT

01-18-2019, 09:15 PM **Post: #10**

**DavidM**
Senior Member

Posts: 722
Joined: Dec 2013

**RE: [VA] SRC#003- New Year 2019 Special**

> **Thomas Klemm Wrote:** ⇒ (01-18-2019 07:04 AM)
>
> This program is for the **HP-11C**:
> ...
> (this may take some time ...)
>
> `12.63898232`

2019/200 completed in about 476 seconds on my 11C.

PM FIND QUOTE REPORT

01-18-2019, 10:28 PM **Post: #11**

**Thomas Klemm**
Senior Member

Posts: 1,449
Joined: Dec 2013

**RE: [VA] SRC#003- New Year 2019 Special**

Looking at the eigenvalues of the matrix $M$:

```
M
EGVL
```

```
[ (175.524449043,0) (-83.049835541,127.396573277)
(-83.049835541,127.396573277) ]
```

respectively rather at their absolute values:

```
[ 175.524449043, 152.076171921, 152.076171921 ]
```

We can estimate the amount of iterations $n$ needed for a 10-digit calculator like the **HP-11C** to return the exact value as:

$$\left(\frac{152.076171921}{175.524449043}\right)^n = 10^{-10}$$

This leads to:

$$n = \frac{-10}{\log_{10}\left(\frac{152.076171921}{175.524449043}\right)} \approx 160.5744$$

Or then for a 12-digit calculator like the **HP-48GX** to:

$$n = \frac{-12}{\log_{10}\left(\frac{152.076171921}{175.524449043}\right)} \approx 192.6892$$

> **DavidM Wrote:** ⇒ (01-18-2019 09:15 PM)
>
> 2019/200 completed in about 476 seconds on my 11C.

Using 160 instead of 200 would take about 380 seconds.

Cheers
Thomas

PM FIND QUOTE REPORT

01-20-2019, 01:33 PM **Post: #12**

**Thomas Klemm**
Senior Member

Posts: 1,449
Joined: Dec 2013

**RE: [VA] SRC#003- New Year 2019 Special**

This program is for the **HP-25**:

**Code:**

```
01: 14 33     :    CLEAR REG
02: 23 04     :    STO 4
03: 22        :    R↓
04: 23 00     :    STO 0
05: 01        :    1
06: 23 03     :    STO 3
07: 24 03     :    RCL 3
08: 24 00     :    RCL 0
09: 61        :    ×
10: 15 73     :    π
11: 23 61 03  :    STO× 3
```

## Example:

```
2019
ENTER↑
200
R/S
12.63898232
```

Cheers
Thomas

PM FIND QUOTE REPORT

01-20-2019, 08:13 PM **Post: #13**

**Thomas Klemm**
Senior Member

Posts: 1,449
Joined: Dec 2013

**RE: [VA] SRC#003- New Year 2019 Special**

This program is for the **HP-29C**:

**Code:**

```
01: 14 33    :     CLEAR REG
02: 23 00    :     STO 0
03: 22       :     R↓
04: 23 04    :     STO 4
05: 01       :     1
06: 23 03    :     STO 3
07: 15 13 00 :    ▸LBL 0
08: 24 03    :     RCL 3
09: 24 04    :     RCL 4
10: 61       :     ×
11: 15 73    :     π
```

## Registers:

```
0: looping counter
1: x
2: y
3: z
4: 2019 (or whatever you entered)
```

## Example:

```
2019
ENTER↑
200
R/S
12.63898232
```

Cheers
Thomas

PM FIND QUOTE REPORT

01-20-2019, 09:40 PM **Post: #14**

**Valentin Albillo**
Senior Member

Posts: 347
Joined: Feb 2015
Warning Level: 0%

**RE: [VA] SRC#003- New Year 2019 Special**

**Hi all:**

First of all, many thanks to all of you who contributed your various RPN/RPL solutions and valuable comments. As Thomas Okken suspected and Thomas Klemm explained, the reason this procedure works and converges to the cubic root of 2019 has all to do with the eigenvalues of the matrix **M**.

My original code for the **HP-71B** (easiest to understand), which exactly follows the steps given in my OP (i.e.: computing the powers of **M** instead of repeatedly multiplying by a vector) is the following *5-line, 144-byte* snippet of code, which also keeps count of the number of iterations needed:

```
1   DESTROY ALL @ OPTION BASE 1 @ DIM M(3,3),B(3,3)
2   DATA PI,2019,2019,1,PI,2019,1,1,PI
3   READ M @ MAT B=M @ R=0 @ I=O
4   REPEAT @ MAT B=B*M @ L=R @ R=B(1,3)/B(2,3)
5   I=I+1 @ UNTIL L=R @ DISP I;R;R^3

>RUN

        183      12.6389823194      2019
```

so after 183 iterations the limit is found to be **12.6389823194**, which is 2019^(1/3), **the cubic root of 2019**. Now for a few comments:

**The procedure can be *generalized* in many ways**. For instance:

**1)** My example used **Pi** in the main diagonal just for aesthetics but actually **the procedure will converge for other positive values K in the main diagonal**, resulting always in the same limit but greatly affecting the number if iterations needed for convergence. For instance:

| K | Iterations |
|---|---|
| 1 | 198 |
| 2 | 200 |
| **Pi** | **183** |
| 10 | 130 |
| 20 | 83 |
| 40 | 51 |
| 80 | 39 |
| **120** | **34** |
| **160** | **33** |
| 200 | 37 |

as you may see in the table above, the value of **K** which results in the lowest number of iterations needed seems to be around 120-140. In fact, the *theoretically optimum* value for **K** in the main diagonal which results in the minimum number of iterations to converge for a given number **N** (2019 in my OP) is:

**`K = (N^(1/3)+N) / (N^(1/3)+1)`**

which for `N=2019` would be

`(2019^(1/3)+2019)/(2019^(1/3)+1) = 148.958253244`

thus placing, say, **148** or **149** in the main diagonal instead of **Pi** will result in the *lowest* number of iterations needed, about *33-34* instead of the *183* needed when **K=Pi**.

**2) The procedure will converge for numbers N other than 2019**, be they integer, real (or even *complex* !), and the limit will be **N^(1/3)**, the cubic root of N. For instance:

- using *K = 1, N = 5:*

$$\mathbf{M} = \begin{matrix} \mathbf{1} & \mathbf{5} & \mathbf{5} \\ \mathbf{1} & \mathbf{1} & \mathbf{5} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} \end{matrix}$$

will converge in 25 iterations to 1.70997594668 = **5^(1/3)**, the cube root of 5

```
      - using K = 2, N = 2:

              2       2       2
      M =     1       2       2
              1       1       2
```

will converge in 14 iterations to 1.25992104989 = **2^(1/3)**, the cube root of 2.

```
      - using K = 1, N = 1 + 2 i: (remember to define M, B, L and R as
COMPLEX)

              1     (1,2)   (1,2)
      M =     1       1     (1,2)
              1       1       1
```

will converge in 23 iterations to 1.21961650797 + 0.471711267789 i = **(1 + 2 i)^(1/3)**, the cube root of 1 + 2 i.

**3)** Though my OP specified the ratio `B(1,3)/B(2,3)`, **the procedure will converge using many other different ratios**. For instance;

```
      B(1,1)/B(2,1), B(2,1)/B(3,1), B(1,2)/B(2,2), B(2,2)/B(3,2), ...,
B(3,3)/B(3,2), B(3,2)/B(3,1)
```

will converge to **N^(1/3)**, the cubic root of N, while the ratios;

```
      B(1,1)/B(3,1), ... , B(3,3)/B(3,1)
```

will converge to **N^(2/3)**, the *square* of the cubic root of N (or the cubic root of the square of N, your choice)

**4)** My OP used a 3x3 matrix and the limit was the cube root of N, but **using DxD matrices** with the same pattern will make the various ratios converge to **N^(1/D)**, N^(2/D), N^(3/D), ..., i.e., **the Dth root of N and its powers**.

**5)** Besides computing cubic roots, the procedure can also be made to converge to **the root of a given cubic *equation*** of a particular form. For instance:

```
      Find a root of:    x^3 - 1.2*x - 2.1 = 0
```

We'll first create the following 3x3 initial matrix **M**:

```
              K       P       Q
      M =     1       K       0
              0       1       K
```

where we'll use K=1, P=1.2, Q=2.1, so the initial matrix will be:

```
              1       1.2     2.1
      M =     1       1       0
              0       1       1
```

which converges in 24 iterations to 1.58816816249, which is the real root of the given cubic equation. This can be checked like this:

```
      >FNROOT(1,2,FVAR^3-1.2*FVAR-2.1)

            1.5881681625
```

**6)** As was the case for *Dth* roots, it's possible to create an initial **DxD matrix** to have the limit converge to **a root of a given polynomial of degree D**. The details are somewhat lengthy and will not be discussed here.

As a final observation, notice the fact that for **integer N and K** this procedure will produce *rational approximations* (integer numerator and denominator) that will converge arbitrarily close to the floating point *real value of the Dth root* using just **integer additions and multiplications** and nothing else: no subtractions, no divisions except the optional one at the end if you wish to convert the fraction to a floating point value. Same for polynomial equations with integer coefficients.

Thanks again to all of you for your interest and valuable contributions, much appreciated, keep them coming ! 😃

Regards.
V.

PM FIND EDIT ✗ QUOTE + REPORT

01-20-2019, 11:28 PM **Post: #15**

**Albert Chan**
Senior Member

Posts: 624
Joined: Jul 2018

**RE: [VA] SRC#003- New Year 2019 Special**

> **Valentin Albillo Wrote:** ⇨ (01-20-2019 09:40 PM)
>
> **2) The procedure will converge for numbers N other than 2019**, be they integer, real (or even *complex* !), and the limit will be **N^(1/3)**, the cubic root of N.

Some N might not converge, say -1

EMAIL PM FIND QUOTE + REPORT

01-27-2019, 11:38 PM **Post: #16**

**Valentin Albillo**
Senior Member

Posts: 347
Joined: Feb 2015
Warning Level: 0%

**RE: [VA] SRC#003- New Year 2019 Special**

> **Albert Chan Wrote:** ⇨ (01-20-2019 11:28 PM)
>
> > **Valentin Albillo Wrote:** ⇨ (01-20-2019 09:40 PM)
> >
> > **2) The procedure will converge for numbers N other than 2019**, be they integer, real (or even *complex* !), and the limit will be **N^(1/3)**, the cubic root of N.
>
> Some N might not converge, say -1

Yes, I know, thanks for your post, Albert.

Funny, I thought that people would post some comments about the 6 generalizations and assorted details I gave in my latest, long, concluding post but a full week has elapsed and save for yours, no one posted a thing or seem to notice. Go figure ... 😕

Regards.
V.
.

PM FIND EDIT ✕ QUOTE REPORT

01-28-2019, 01:56 PM (This post was last modified: 01-28-2019 01:58 PM by pier4r.) **Post: #17**

**pier4r**
Senior Member

Posts: 1,963
Joined: Nov 2014

**RE: [VA] SRC#003- New Year 2019 Special**

I appreciate the content. I though Thomas was covering most of it but instead there were many more patterns and properties.

Then again, the less trivial the content, the less the chance to discuss it due to the needed effort to understand it. https://en.wikipedia.org/wiki/Law_of_triviality

*"The time spent on any item of the agenda will be in inverse proportion to the sum [of money] involved."*

For technical discussions one can change it so
*"The time spent on any item of the agenda will be in inverse proportion to the knowledge and understanding effort involved."*

---

Wikis are great, Contribute :)

EMAIL PM FIND QUOTE REPORT

01-28-2019, 02:46 PM (This post was last modified: 01-28-2019 02:57 PM by Albert Chan.) **Post: #18**

**Albert Chan**
Senior Member

Posts: 624
Joined: Jul 2018

**RE: [VA] SRC#003- New Year 2019 Special**

> **pier4r Wrote:** ⇒ (01-28-2019 01:56 PM)
>
> *"The time spent on any item of the agenda will be in inverse proportion to the knowledge and understanding effort involved."*

+1
I PM Thomas last week for how his estimated iterations work.
Sadly, I still don't understand the geometric intuition ...

> **Thomas Klemm Wrote:**
>
> > **Albert Chan Wrote:**
> >
> > How is above formula derived ?
>
> If the matrix $M$ is diagonalized the powers can be calculated easily as:

$$\begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix}^n = \begin{bmatrix} \lambda_1^n & 0 & 0 \\ 0 & \lambda_2^n & 0 \\ 0 & 0 & \lambda_3^n \end{bmatrix}$$

Thus assuming the initial vector $v$ in a general position the ratio of the 2nd to the 1st coordinate after iterating $n$ times will be roughly:

$$\frac{\lambda_2^n}{\lambda_1^n}$$

We want that to be so small that it doesn't affect the display:

$$1 + 10^{-10} = 1$$

For the given example I have a geometric intuition.
A vector is stretched in direction of the 1st eigenvector by factor 175.524449043.
The other two eigenvectors define a plane where the projection upon is rotated and stretched by 152.076171921.
Now you can wonder how often you have to iterate such that stretching in the dominant direction is so much bigger that the rest becomes irrelevant.

I assumed you are familiar with the concepts of eigenvalue and eigenvectors from linear algebra.
Otherwise it's not possible to explain the formula.

HTH
Thomas

EMAIL PM FIND QUOTE REPORT

01-28-2019, 07:40 PM **Post: #19**

**Thomas Klemm**
Senior Member

Posts: 1,449
Joined: Dec 2013

**RE: [VA] SRC#003- New Year 2019 Special**

> **Albert Chan Wrote:** (01-28-2019 02:46 PM)
>
> Sadly, I still don't understand the geometric intuition ...

Sorry about that.

Let's start with a vector

$$v = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

and a matrix

$$M = \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix}$$

If we calculate $M^n v$ we end up with:

$$M^n v = \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix}^n \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 3^n & 0 \\ 0 & 2^n \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 3^n \\ 2^n \end{bmatrix}$$

Thus the ratio of the coordinates is:

$$\frac{y}{x} = \frac{2^n}{3^n} = \left(\frac{2}{3}\right)^n$$

If we want to know when the $y$ component can be neglected compared to the $x$ component using a 12-digit calculator we see that's when:

$$\frac{y}{x} < 10^{-12}$$

And therefore when:

$$\left(\frac{2}{3}\right)^n < 10^{-12}$$

Solve that for $n$ to get:

$$n > \frac{\log_{10} 10^{-12}}{\log_{10} \frac{2}{3}} = \frac{-12}{\log_{10} 2 - \log_{10} 3} \approx 68.1465$$

Indeed:

```
2 ENTER 3 ÷ 69 yˣ
7.0746e-13
```

But of course we could get a bit closer by using *5e-12* instead of *1e-12*.
This leads to $n > 64.1771$ and indeed $n = 65$ is enough:

```
2 ENTER 3 ÷ 65 yˣ
3.5815e-12
```

---

Using the eigenvectors as a basis just adds some coordinate transformations.
We don't know the details (or rather I ignored them) but for a general vector that shouldn't matter much.

---

But of course this calculation changes if we start with something like:

$$v = \begin{bmatrix} 1 \\ 10^k \end{bmatrix}$$

for a big positive value of $k$.
But even that would just add $k$ to the result.

That's why I consider it only a rough estimate.

Does this make more sense?

Cheers
Thomas

PM FIND QUOTE REPORT

01-28-2019, 08:17 PM **Post: #20**

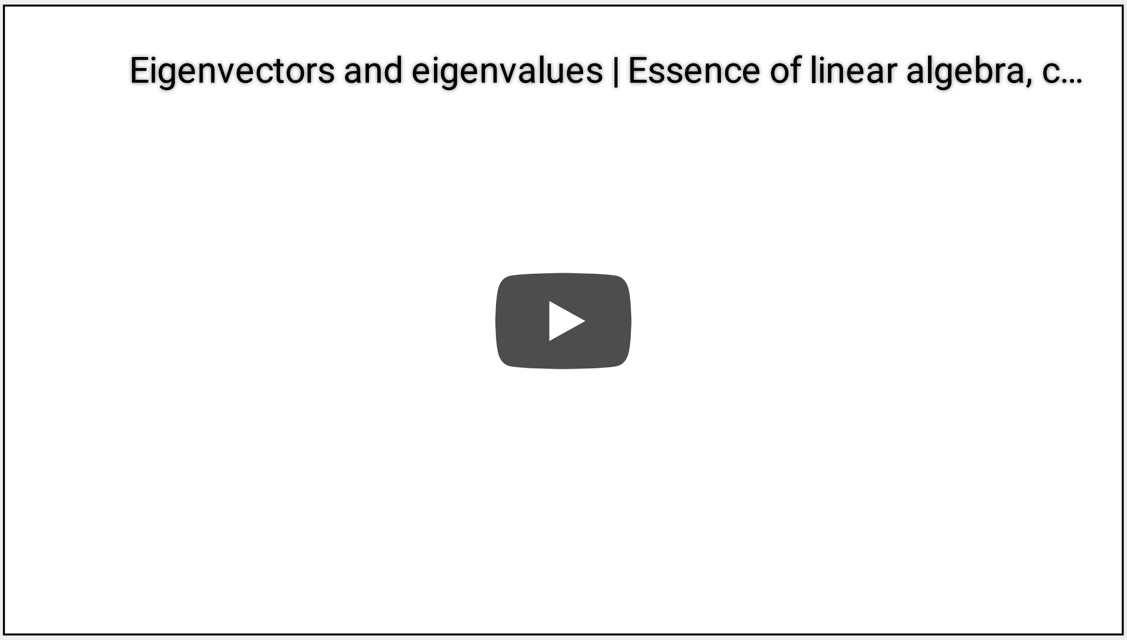
**Thomas Klemm** Posts: 1,449

Senior Member Joined: Dec 2013

**RE: [VA] SRC#003- New Year 2019 Special**

> **Thomas Klemm Wrote:**
>
> I assumed you are familiar with the concepts of eigenvalue and eigenvectors from linear algebra.

Maybe this video can give some geometric intuition:


Eigenvectors and eigenvalues | Essence of linear algebra, c...

PM FIND QUOTE REPORT

« Next Oldest | Next Newest » Enter Keywords Search Thread

Pages (2): 1 2 Next » NEW REPLY

View a Printable Version

Send this Thread to a Friend

Subscribe to this thread

User(s) browsing this thread: Valentin Albillo*

Contact Us | The Museum of HP Calculators | Return to Top | Return to Content | Lite (Archive) Mode | RSS Syndication English (American) ▼ Go