Homework 4

Alejandra Torres Manotas

March 4, 2019

1 Python Performance

Kernprof, Line Profiler and Memory Profiler

- What is that code doing and where?
- Where is it spending the majority of the time?
- How much memory is being used
- where is the most used one?

Here I have the line profile of V1.py. This time measure was made creatting two functions (main1, and main2) which let me to study the line profile in the code.

In the first picture you can see the line profile of the $main1()$ function (Figure 1), with a runtime of $0.0054s$ and $525.7s$, respectively. Here we iterated 1000 times the $HenonMap()$ functon, and we created two vectors of (x, y)-values with the inicial conditions given from line 11 to line 19. Then, we create two vectors and created a foor-loop which called the HenonMap function the number of iterations given before. And the major of time was spent in the HenonMap evaluation.

In the second picture you can see the line profile of the $main2()$ function (Figure 2). Here in the lines $(38 - 46)$ was defined the initial values of variables and constant values, in the lines $(48 - 51)$ was defined the size of the grid, and the matrix. Finally, in the lines $(53 - 63)$ was developed the HenonMap evaluation on the grid with the condition $|p| < R$. The major time of running was spent in the Henon map evaluation and in the While conditional quiestion.

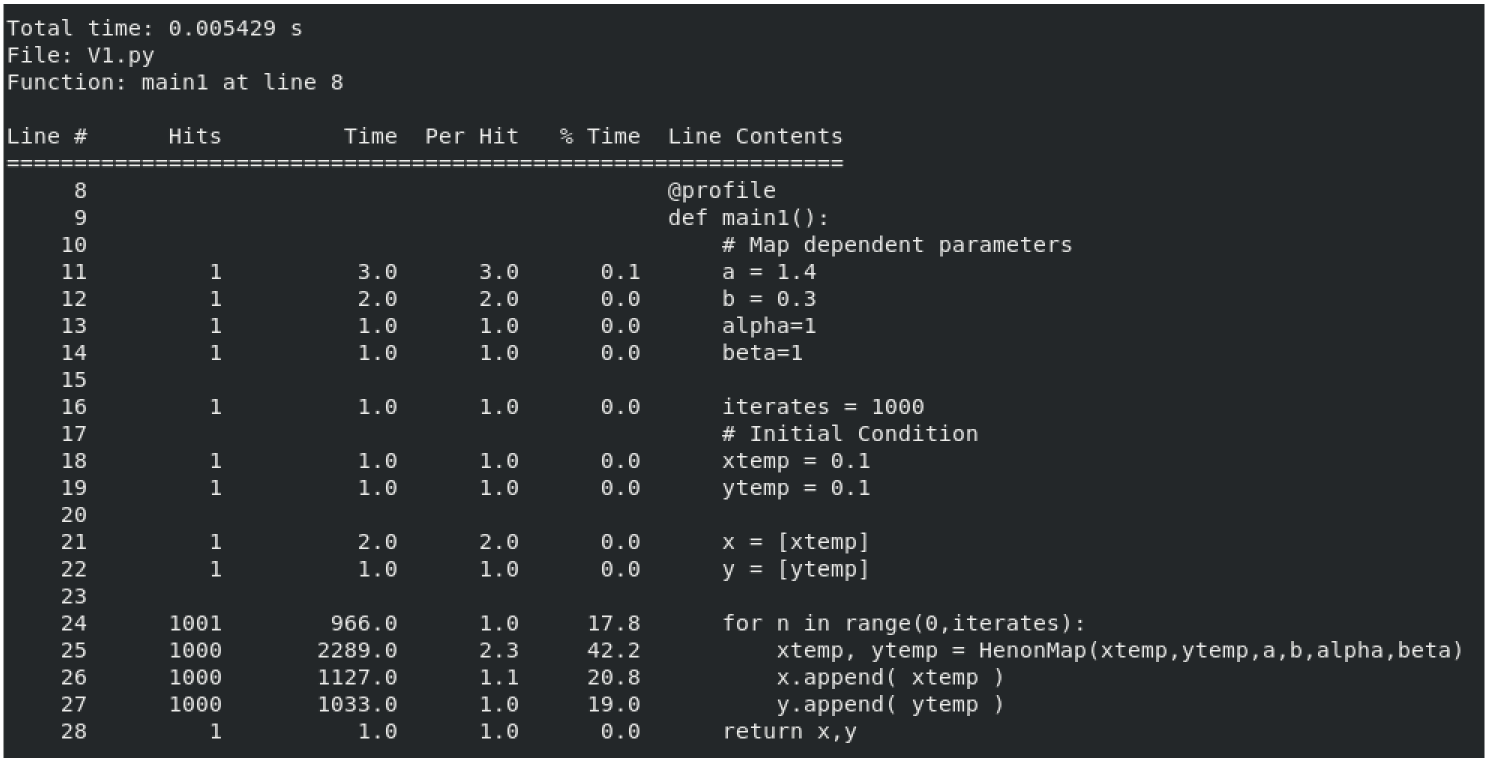
Finally, the memory-profile for the main1 function (Figure 3) said that the momory used was 118.4 MiB, and in the main2 (Figure 2) the memory used was 145.8 MiB.

Not Numpy

In the file V2.py the numpy parts, that is, the linespace and the creation of the matrix was changed by normal for-loops and comprehension for-loops. As it can be seen in the Figure (5) the total runtime in the part changed was 1000-times more.

Optimise

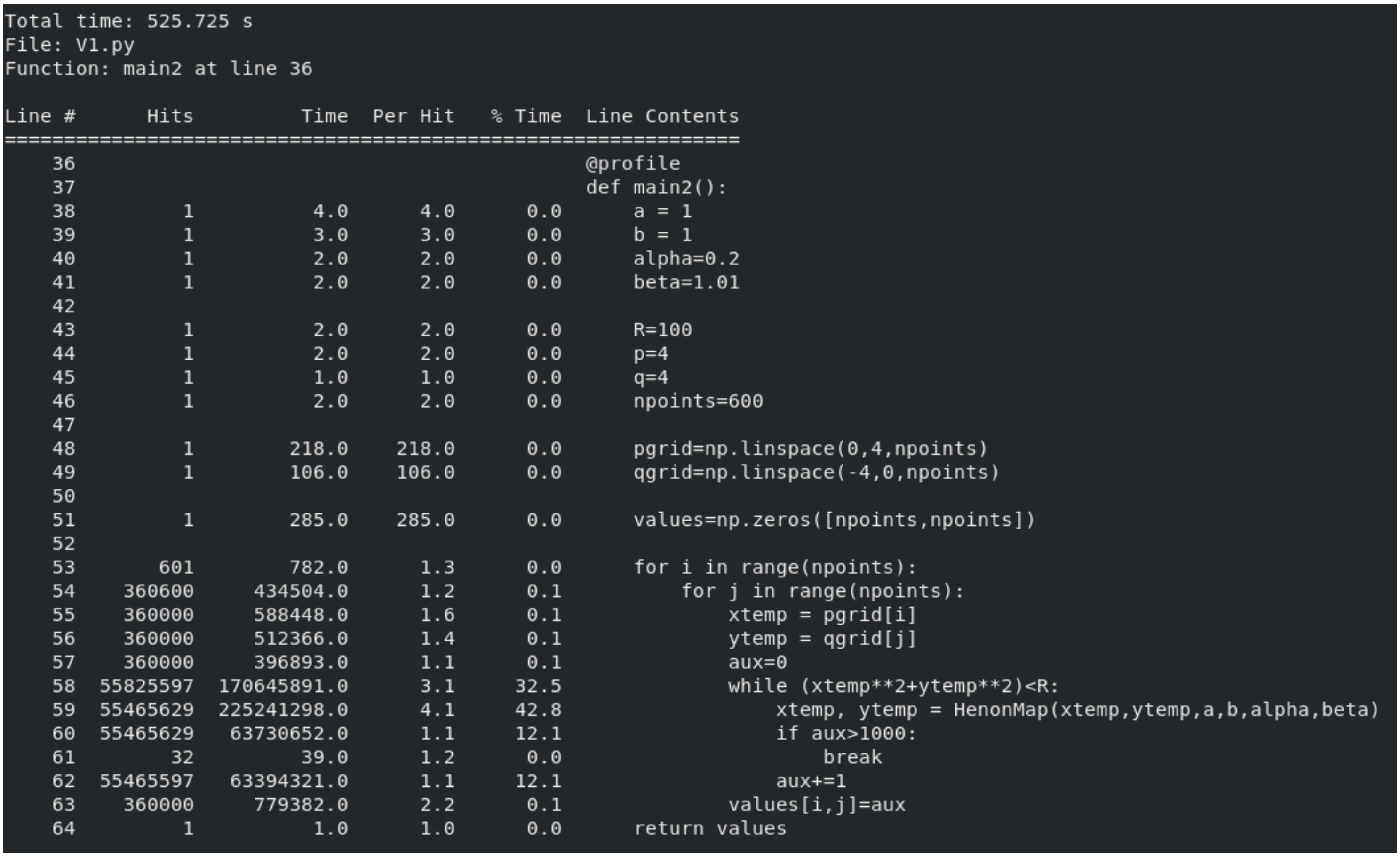
In this part was optimised the code V1.py on the Jupyther Notebook with the name V3.ipnyb with the used of $@jit((float64, ..))$ over the HenonMap, $@jit(parallel = True)$ over the main2(), and with $@jit$ over the main1(). In the main1() the runtime was 51 times less, and in the main2() the runtime was 2468 times less.



```
Total time: 0.005429 s
File: V1.py
Function: main1 at line 8

Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
     8                                           @profile
     9                                           def main1():
    10                                               # Map dependent parameters
    11         1          3.0      3.0      0.1      a = 1.4
    12         1          2.0      2.0      0.0      b = 0.3
    13         1          1.0      1.0      0.0      alpha=1
    14         1          1.0      1.0      0.0      beta=1
    15
    16         1          1.0      1.0      0.0      iterates = 1000
    17                                               # Initial Condition
    18         1          1.0      1.0      0.0      xtemp = 0.1
    19         1          1.0      1.0      0.0      ytemp = 0.1
    20
    21         1          2.0      2.0      0.0      x = [xtemp]
    22         1          1.0      1.0      0.0      y = [ytemp]
    23
    24      1001        966.0      1.0     17.8      for n in range(0,iterates):
    25      1000       2289.0      2.3     42.2          xtemp, ytemp = HenonMap(xtemp,ytemp,a,b,alpha,beta)
    26      1000       1127.0      1.1     20.8          x.append( xtemp )
    27      1000       1033.0      1.0     19.0          y.append( ytemp )
    28         1          1.0      1.0      0.0      return x,y
```

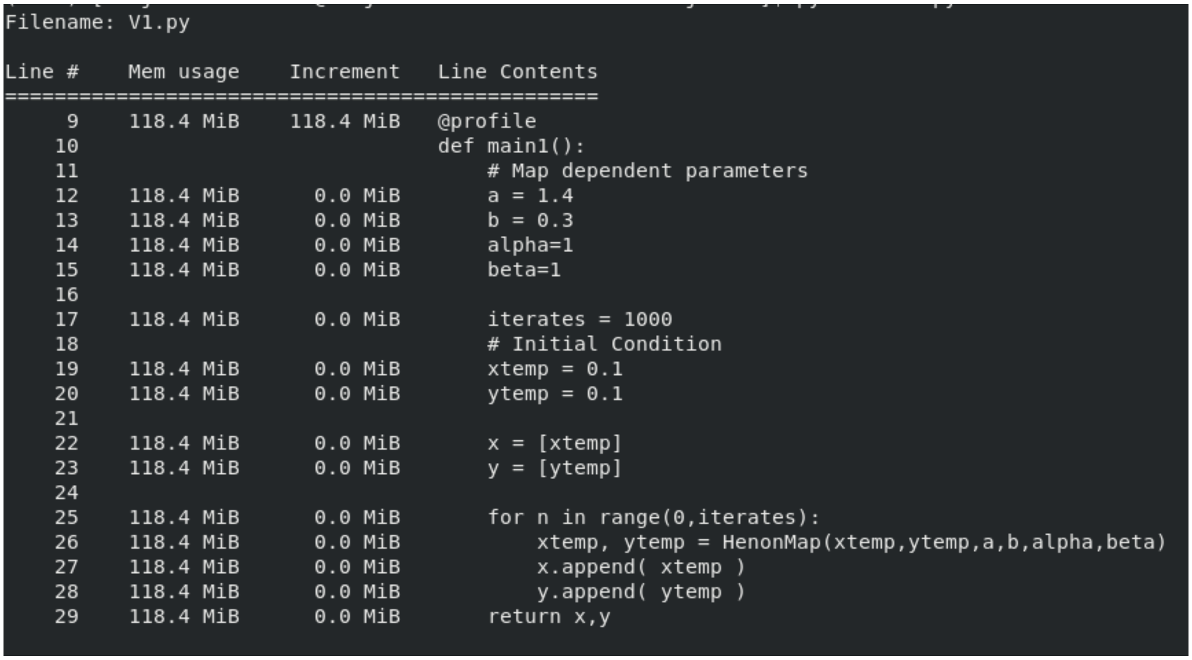
Figure 1: Line-profiler of V1 file in $main1()$ function



```
Total time: 525.725 s
File: V1.py
Function: main2 at line 36

Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
    36                                           @profile
    37                                           def main2():
    38         1          4.0      4.0      0.0      a = 1
    39         1          3.0      3.0      0.0      b = 1
    40         1          2.0      2.0      0.0      alpha=0.2
    41         1          2.0      2.0      0.0      beta=1.01
    42
    43         1          2.0      2.0      0.0      R=100
    44         1          2.0      2.0      0.0      p=4
    45         1          1.0      1.0      0.0      q=4
    46         1          2.0      2.0      0.0      npoints=600
    47
    48         1        218.0    218.0      0.0      pgrid=np.linspace(0,4,npoints)
    49         1        106.0    106.0      0.0      qgrid=np.linspace(-4,0,npoints)
    50
    51         1        285.0    285.0      0.0      values=np.zeros([npoints,npoints])
    52
    53       601        782.0      1.3      0.0      for i in range(npoints):
    54    360600     434504.0      1.2      0.1          for j in range(npoints):
    55    360000     588448.0      1.6      0.1              xtemp = pgrid[i]
    56    360000     512366.0      1.4      0.1              ytemp = qgrid[j]
    57    360000     396893.0      1.1      0.1              aux=0
    58  55825597  170645891.0      3.1     32.5              while (xtemp**2+ytemp**2)<R:
    59  55465629  225241298.0      4.1     42.8                  xtemp, ytemp = HenonMap(xtemp,ytemp,a,b,alpha,beta)
    60  55465629   63730652.0      1.1     12.1                  if aux>1000:
    61        32         39.0      1.2      0.0                      break
    62  55465597   63394321.0      1.1     12.1                  aux+=1
    63    360000     779382.0      2.2      0.1              values[i,j]=aux
    64         1          1.0      1.0      0.0      return values
```

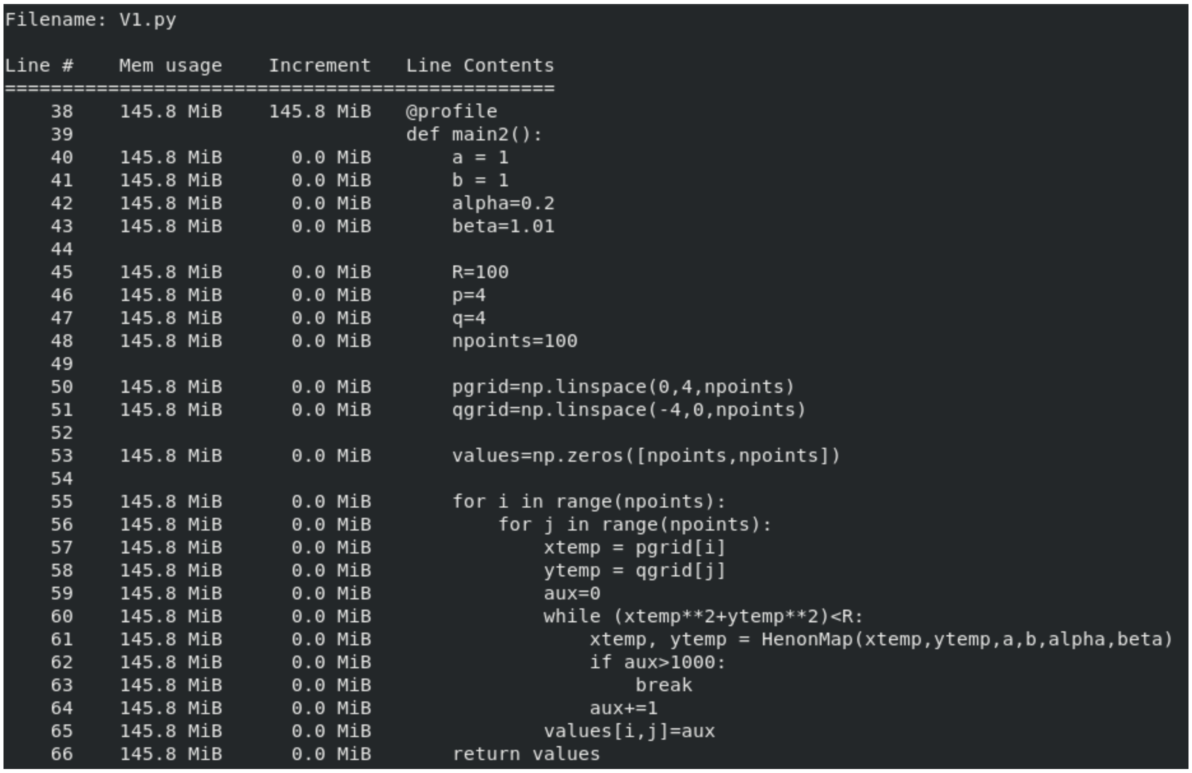
Figure 2: Line-profiler of V1 file in *main*2() function



```
Filename: V1.py

Line #    Mem usage    Increment   Line Contents
================================================
     9    118.4 MiB    118.4 MiB   @profile
    10                             def main1():
    11                                 # Map dependent parameters
    12    118.4 MiB      0.0 MiB       a = 1.4
    13    118.4 MiB      0.0 MiB       b = 0.3
    14    118.4 MiB      0.0 MiB       alpha=1
    15    118.4 MiB      0.0 MiB       beta=1
    16
    17    118.4 MiB      0.0 MiB       iterates = 1000
    18                                 # Initial Condition
    19    118.4 MiB      0.0 MiB       xtemp = 0.1
    20    118.4 MiB      0.0 MiB       ytemp = 0.1
    21
    22    118.4 MiB      0.0 MiB       x = [xtemp]
    23    118.4 MiB      0.0 MiB       y = [ytemp]
    24
    25    118.4 MiB      0.0 MiB       for n in range(0,iterates):
    26    118.4 MiB      0.0 MiB           xtemp, ytemp = HenonMap(xtemp,ytemp,a,b,alpha,beta)
    27    118.4 MiB      0.0 MiB           x.append( xtemp )
    28    118.4 MiB      0.0 MiB           y.append( ytemp )
    29    118.4 MiB      0.0 MiB       return x,y
```

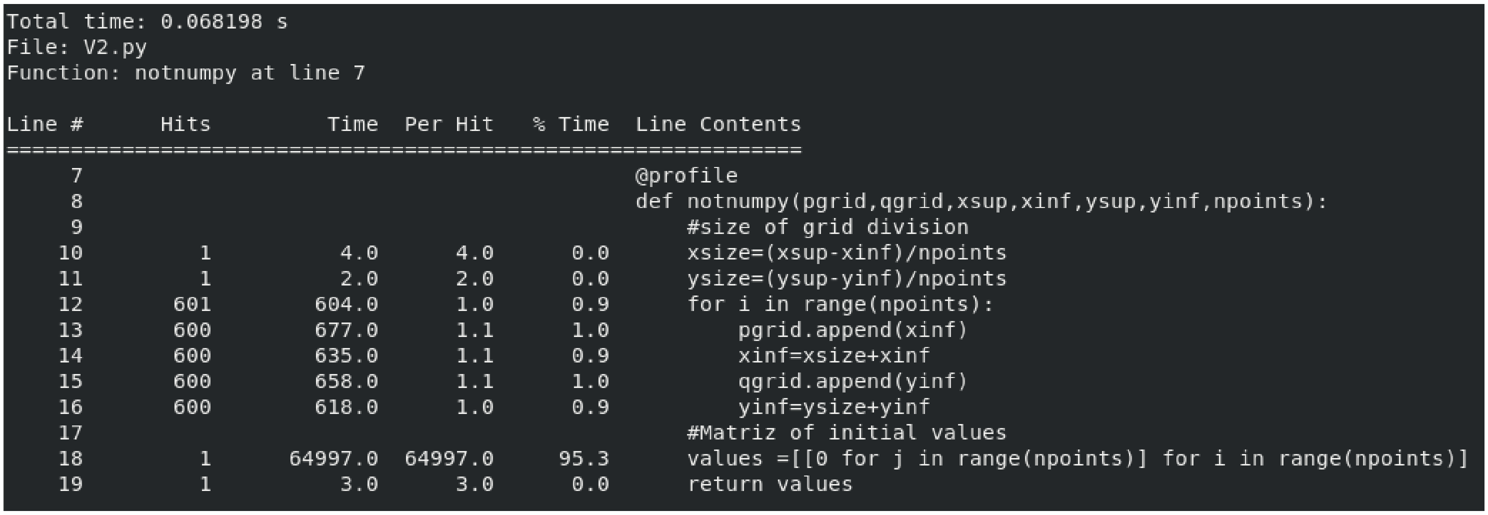
Figure 3: Memory-profiler of V1 file in *main*1() function



```
Filename: V1.py

Line #    Mem usage    Increment   Line Contents
================================================
    38    145.8 MiB    145.8 MiB   @profile
    39                             def main2():
    40    145.8 MiB      0.0 MiB       a = 1
    41    145.8 MiB      0.0 MiB       b = 1
    42    145.8 MiB      0.0 MiB       alpha=0.2
    43    145.8 MiB      0.0 MiB       beta=1.01
    44
    45    145.8 MiB      0.0 MiB       R=100
    46    145.8 MiB      0.0 MiB       p=4
    47    145.8 MiB      0.0 MiB       q=4
    48    145.8 MiB      0.0 MiB       npoints=100
    49
    50    145.8 MiB      0.0 MiB       pgrid=np.linspace(0,4,npoints)
    51    145.8 MiB      0.0 MiB       qgrid=np.linspace(-4,0,npoints)
    52
    53    145.8 MiB      0.0 MiB       values=np.zeros([npoints,npoints])
    54
    55    145.8 MiB      0.0 MiB       for i in range(npoints):
    56    145.8 MiB      0.0 MiB           for j in range(npoints):
    57    145.8 MiB      0.0 MiB               xtemp = pgrid[i]
    58    145.8 MiB      0.0 MiB               ytemp = qgrid[j]
    59    145.8 MiB      0.0 MiB               aux=0
    60    145.8 MiB      0.0 MiB               while (xtemp**2+ytemp**2)<R:
    61    145.8 MiB      0.0 MiB                   xtemp, ytemp = HenonMap(xtemp,ytemp,a,b,alpha,beta)
    62    145.8 MiB      0.0 MiB                   if aux>1000:
    63    145.8 MiB      0.0 MiB                       break
    64    145.8 MiB      0.0 MiB                   aux+=1
    65    145.8 MiB      0.0 MiB               values[i,j]=aux
    66    145.8 MiB      0.0 MiB       return values
```

Figure 4: Memory-profiler of V1 file in *main2*() function



```
Total time: 0.068198 s
File: V2.py
Function: notnumpy at line 7

Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
     7                                           @profile
     8                                           def notnumpy(pgrid,qgrid,xsup,xinf,ysup,yinf,npoints):
     9                                               #size of grid division
    10         1          4.0      4.0      0.0      xsize=(xsup-xinf)/npoints
    11         1          2.0      2.0      0.0      ysize=(ysup-yinf)/npoints
    12       601        604.0      1.0      0.9      for i in range(npoints):
    13       600        677.0      1.1      1.0          pgrid.append(xinf)
    14       600        635.0      1.1      0.9          xinf=xsize+xinf
    15       600        658.0      1.1      1.0          qgrid.append(yinf)
    16       600        618.0      1.0      0.9          yinf=ysize+yinf
    17                                               #Matriz of initial values
    18         1      64997.0  64997.0     95.3      values =[[0 for j in range(npoints)] for i in range(npoints)]
    19         1          3.0      3.0      0.0      return values
```

Figure 5: Line-profiler of V2 file in the changed numpy part