

1 Tutorial: Running Monoids in Jupyter Notebook with Python 3.0 and SageMath for End-Product Inhibition Cases

Author: Marco Polo Castillo Villalba.

Step 1: Install a Jupyter Notebook Platform

Follow the steps below based on your operating system (Windows/macOS/Linux):

1. Visit: <https://www.anaconda.com/products/distribution>
2. Download the appropriate installer:
 - Windows: `.exe`
 - macOS: `.pkg`
 - Linux: `.sh`
3. Run the installer and follow the setup instructions (default settings are recommended).
4. After installation:
 - Open Anaconda Navigator and click "Launch" under Jupyter Notebook, OR
 - Open a terminal (or Anaconda Prompt) and run: `jupyter notebook`. On Ubuntu: `jupyter-notebook`.

Step 2: Follow the Screenshots to Run Monoids for End-Product Inhibition Cases

Note: The script supports monoid generation for cones of dimension $n = 1$ up to $n = 9$. Use the correct filenames for each case. For example: `Cone3n_1.txt`, `Cone3n_2.txt`, ..., `Cone9n_10.txt`.

Note: To concatenate all the individual output files into one, use the following command format in Linux:

```
cat *_TF_DNA31.sage >> OUT31_TF_DNAsage.txt
```

Replace 31 with the corresponding number for your case. For $n = 4$, you would run:

```
cat *_TF_DNA41.sage >> OUT41_TF_DNAsage.txt
cat *_TF_DNA42.sage >> OUT42_TF_DNAsage.txt
cat *_TF_DNA43.sage >> OUT43_TF_DNAsage.txt
cat *_TF_DNA44.sage >> OUT44_TF_DNAsage.txt
cat *_TF_DNA45.sage >> OUT45_TF_DNAsage.txt
```

Case for the length of the pathway $n=7$. Allosteric system for end-product inhibition (lysine-aspartate).

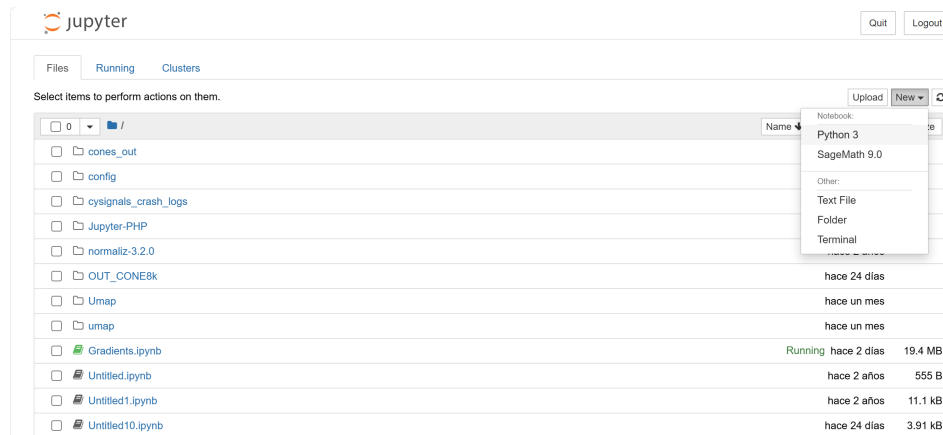


Figure 1: Open a new notebook using Python 3.

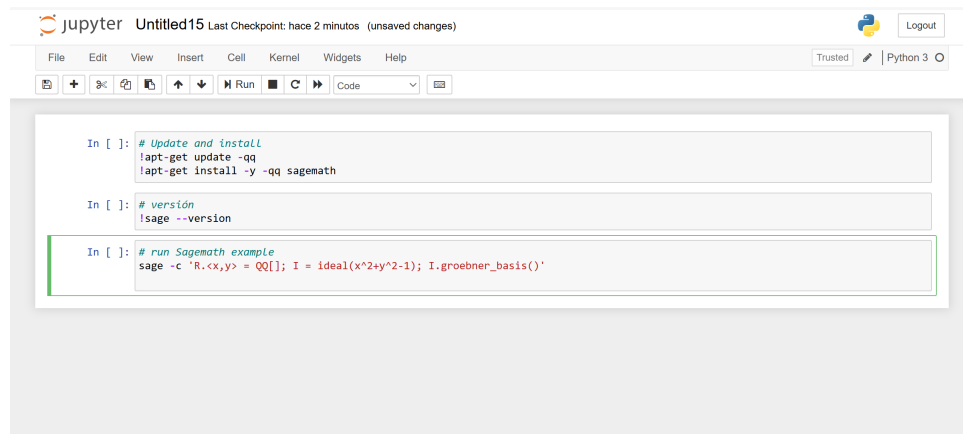


Figure 2: Install the platform needed to run algorithms in computational algebraic geometry. For computing monoid generators, copy and paste the code shown in this figure into your notebook.

```

1  #!/usr/bin/env python3
2  import numpy as np
3  from scipy.stats import boltzmann
4  import argparse
5  import os
6
7  def apply_fixed_zero_structure(A, zero_patterns):
8      for row, cols in zero_patterns:
9          for col in cols:
10             A[row][col] = 0
11     return A
12
13
14  def Cone(n, lambda_, N, f):
15     filename = os.path.basename(f)
16
17     # Define zeroing patterns based on file name
18     zero_patterns = {
19         "Cone9n_1.txt": [
20             (0, [2, 3, 4, 5, 6, 7]),
21             (1, [2, 3, 4, 5, 6]),
22             (2, [1, 2, 3, 4, 5, 6]),
23             (3, [1, 2, 3, 4, 5, 6]),

```

Figure 3: Open the script file `Generator_of_cones.py`, which is available in the GitHub repository. Copy and paste it into a new Python notebook. You can write the Linux line: `python3 Generator_of_cones.py -n 500 -f Cone3n_1.txt`



Figure 4: Once SageMath version 9.0 is installed, open a new notebook in SageMath.

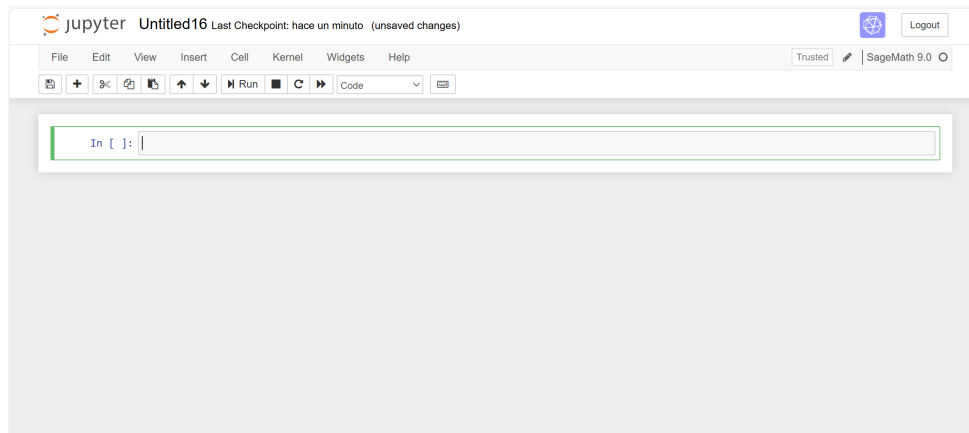


Figure 5: A new SageMath notebook will open. Leave it open; you'll run a script in this notebook shortly.

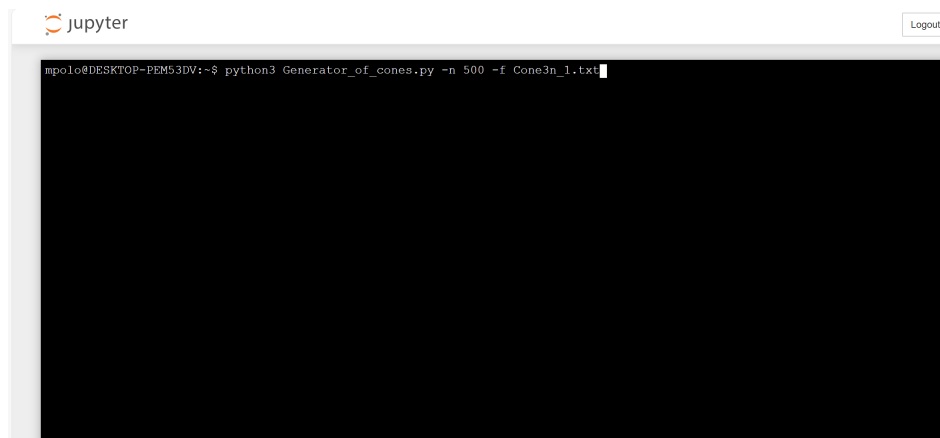
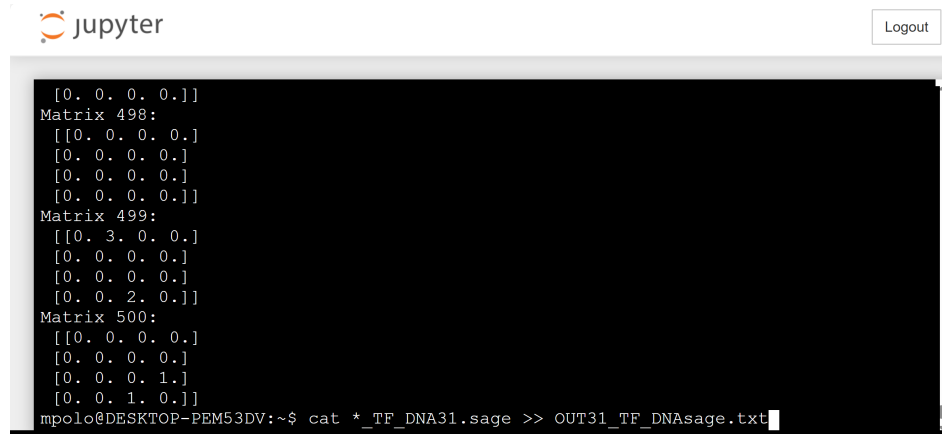


Figure 6: Open a new terminal in Jupyter. Type the command shown in the figure. File names must follow the naming convention: **Cone3n_1.txt**, **Cone3n_2.txt**, ..., **Cone7n_8.txt**, depending on the length n of the pathways. The **-n 500** option generates 500 random matrices using the Boltzmann distribution.

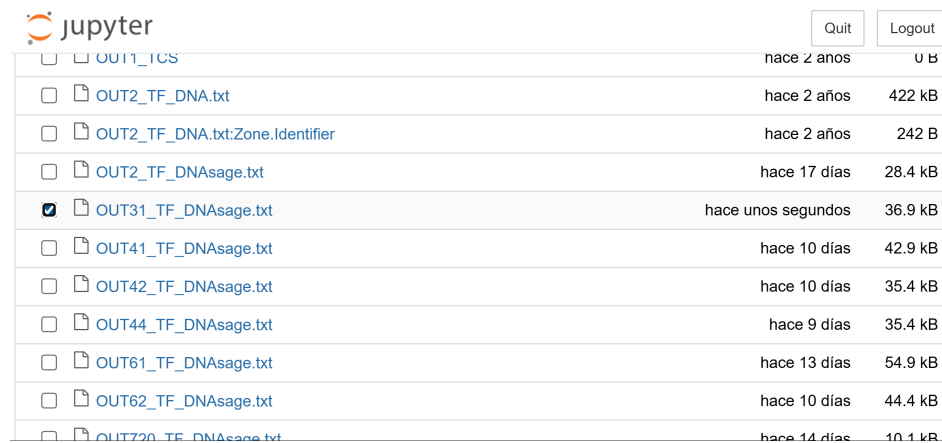


```

[0. 0. 0. 0.]
Matrix 498:
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
Matrix 499:
[[0. 3. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 2. 0.]]
Matrix 500:
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 1.]
 [0. 0. 1. 0.]]
mpolo@DESKTOP-PEM53DV:~$ cat *_TF_DNA31.sage >> OUT31_TF_DNA.sage.txt

```

Figure 7: After generating 500 random matrices, save them in a single file using the command shown. The output file in this case is `OUT31.TF_DNA.sage.txt`.



jupyter		Quit	Logout
<input type="checkbox"/>	OUT1_TCS	hace 2 años	U B
<input type="checkbox"/>	OUT2_TF_DNA.txt	hace 2 años	422 kB
<input type="checkbox"/>	OUT2_TF_DNA.txt:Zone.Identifier	hace 2 años	242 B
<input type="checkbox"/>	OUT2_TF_DNA.sage.txt	hace 17 días	28.4 kB
<input checked="" type="checkbox"/>	OUT31_TF_DNA.sage.txt	hace unos segundos	36.9 kB
<input type="checkbox"/>	OUT41_TF_DNA.sage.txt	hace 10 días	42.9 kB
<input type="checkbox"/>	OUT42_TF_DNA.sage.txt	hace 10 días	35.4 kB
<input type="checkbox"/>	OUT44_TF_DNA.sage.txt	hace 9 días	35.4 kB
<input type="checkbox"/>	OUT61_TF_DNA.sage.txt	hace 13 días	54.9 kB
<input type="checkbox"/>	OUT62_TF_DNA.sage.txt	hace 10 días	44.4 kB
<input type="checkbox"/>	OUT720_TF_DNA.sage.txt	hace 14 días	10.1 kB

Figure 8: You can verify the saved output file in the Jupyter file tree as shown.

```

471 H73 = Matrix(ZZ, 4, 4, [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0])
472 H74 = Matrix(ZZ, 4, 4, [0, 2, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0])
473 H75 = Matrix(ZZ, 4, 4, [0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 1, 0, 0, 0, 0, 0])
474 H76 = Matrix(ZZ, 4, 4, [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0])
475 H77 = Matrix(ZZ, 4, 4, [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0])
476 H78 = Matrix(ZZ, 4, 4, [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0])
477 H79 = Matrix(ZZ, 4, 4, [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
478 H7 = Matrix(ZZ, 4, 4, [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0])
479 H80 = Matrix(ZZ, 4, 4, [0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0])
480 H81 = Matrix(ZZ, 4, 4, [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0])
481 H82 = Matrix(ZZ, 4, 4, [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
482 H83 = Matrix(ZZ, 4, 4, [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0])
483 H84 = Matrix(ZZ, 4, 4, [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
484 H85 = Matrix(ZZ, 4, 4, [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0])
485 H86 = Matrix(ZZ, 4, 4, [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
486 H87 = Matrix(ZZ, 4, 4, [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0])
487 H88 = Matrix(ZZ, 4, 4, [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
488 H89 = Matrix(ZZ, 4, 4, [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0])
489 H8 = Matrix(ZZ, 4, 4, [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0])
490 H90 = Matrix(ZZ, 4, 4, [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
491 H91 = Matrix(ZZ, 4, 4, [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0])
492 H92 = Matrix(ZZ, 4, 4, [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0])
493 H93 = Matrix(ZZ, 4, 4, [0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 2, 0, 0, 0, 0, 0, 0])
494 H94 = Matrix(ZZ, 4, 4, [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
495 H95 = Matrix(ZZ, 4, 4, [0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0])
496 H96 = Matrix(ZZ, 4, 4, [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0])

```

Figure 9: The content of the output file should appear as shown. These files are formatted for execution in SageMath.

```

In [ ]: # File path (change this if needed)
file_path = "OUT31_TF_DNAAsage.txt"

# File path - update this to your file location

# Safe Local namespace to store matrices
namespace = {}

# Use Sage globals so 'Matrix', 'ZZ' are defined
with open(file_path, "r") as f:
    for line in f:
        line = line.strip()
        if line:
            exec(line, globals(), namespace) # ~ FIXED: pass Sage globals here

# Collect matrices, sorted by index (e.g. H1, H2, ..., H28)
L = [namespace[k] for k in sorted(namespace, key=lambda x: int(x[1:]))]

matrix_dict = {key: namespace[key] for key in sorted(namespace) if key.startswith("H")}
matrix_names = list(matrix_dict.keys())
print(eval(f"matrix_names[300]"))
print(matrix_dict[eval(f"matrix_names[277]")])
N = matrix_dict[eval(f"matrix_names[300]")]
print(N.nrows())
# Example usage

```

Figure 10: Return to the SageMath notebook left open in Figure 5. Copy and paste the script from the file `Sagemathscript.txt` (available in the GitHub repository). Make sure to set the filename at the beginning of the script (e.g., `OUT31_TF_DNAAsage.txt`).

```

print(B)
#break

--- Processing cone 441 ---
[ 1  0  0  0]
[ 0  0  0  1]
[ 0  1  0  0]
[ 0 -1  0  0]

--- Processing cone 442 ---
[ 0  1  0  0]
[ 1  0  0  0]
[-1  0  0  0]
[ 0  0  1  0]
[ 1  0  0  0]
[-1  0  0  0]
[ 0  0  1  0]
[ 0  0 -1  0]

--- Processing cone 443 ---
[ 1  0  0  0]
[-1  0  0  0]

```

Figure 11: After pasting the script into the notebook, run it. The output will be 500 matrix representations of monoids for the selected pathway length n .

```

[0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 1.]
Matrix 496:
[[0. 0. 0. 0. 0. 0. 0. 1. 0.]
[0. 0. 0. 0. 0. 1. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0.]
[1. 0. 0. 0. 0. 0. 0. 0.]]
Matrix 497:
[[0. 1. 0. 0. 0. 0. 0. 2.]
[0. 0. 0. 0. 0. 0. 0. 1.]
[2. 0. 0. 0. 0. 0. 0. 1.]
[0. 0. 0. 0. 0. 0. 0. 1.]]
Matrix 498:
[[0. 0. 0. 0. 0. 0. 0. 1.]
[0. 0. 0. 0. 0. 1. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 1. 0.]]
Matrix 499:
[[0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0.]
[1. 0. 0. 0. 0. 0. 0. 0.]
[1. 0. 0. 0. 0. 0. 0. 0.]]
Matrix 500:
[[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. 0. 1. 0.]]
mpolo@DESKTOP-PEM53DV:~$ cat *_TF_DNA7*.sage >> OUT71_TF_DNA7sage.txt

```

Figure 12: You only write in new terminal on you Jupyter-Notebook the line in linux: `python3 Generator_of_cones.py -n 500 -f Cone7n.1.txt`



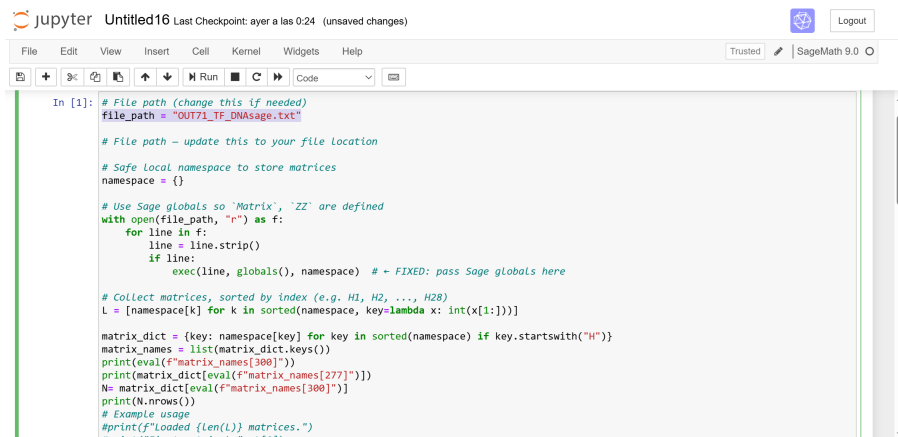
<input type="checkbox"/>	152_TF_DNA71.sage	hace 2 minutos	122 B
<input type="checkbox"/>	153_TF_DNA71.sage	hace 2 minutos	122 B
<input type="checkbox"/>	154_TF_DNA71.sage	hace 2 minutos	122 B
<input type="checkbox"/>	155_TF_DNA71.sage	hace 2 minutos	122 B
<input type="checkbox"/>	156_TF_DNA71.sage	hace 2 minutos	122 B
<input type="checkbox"/>	157_TF_DNA71.sage	hace 2 minutos	122 B
<input type="checkbox"/>	158_TF_DNA71.sage	hace 2 minutos	122 B
<input type="checkbox"/>	159_TF_DNA71.sage	hace 2 minutos	122 B
<input type="checkbox"/>	15_TF_DNA71.sage	hace 2 minutos	121 B
<input type="checkbox"/>	160_TF_DNA71.sage	hace 2 minutos	122 B
<input type="checkbox"/>	161_TF_DNA71.sage	hace 2 minutos	122 B
<input type="checkbox"/>	162_TF_DNA71.sage	hace 2 minutos	122 B
<input type="checkbox"/>	163_TF_DNA71.sage	hace 2 minutos	122 B
<input type="checkbox"/>	164_TF_DNA71.sage	hace 2 minutos	122 B
<input type="checkbox"/>	165_TF_DNA71.sage	hace 2 minutos	122 B
<input type="checkbox"/>	166_TF_DNA71.sage	hace 2 minutos	122 B
<input type="checkbox"/>	167_TF_DNA71.sage	hace 2 minutos	122 B

Figure 13: After write the following liux line to get your files in an only file: `cat *_TF_DNA71.sage >> OUT71_TF_DNA71.txt`



<input type="checkbox"/>	OUT2_TF_DNA71.txt	hace 19 dias	28.4 kB
<input type="checkbox"/>	OUT31_TF_DNA71.txt	hace 2 dias	36.9 kB
<input type="checkbox"/>	OUT41_TF_DNA71.txt	hace 12 dias	42.9 kB
<input type="checkbox"/>	OUT42_TF_DNA71.txt	hace 12 dias	35.4 kB
<input type="checkbox"/>	OUT44_TF_DNA71.txt	hace 11 dias	35.4 kB
<input type="checkbox"/>	OUT61_TF_DNA71.txt	hace 15 dias	54.9 kB
<input type="checkbox"/>	OUT62_TF_DNA71.txt	hace 12 dias	44.4 kB
<input checked="" type="checkbox"/>	OUT71_TF_DNA71.txt	hace 2 minutos	60.9 kB
<input type="checkbox"/>	OUT81_TF_DNA71.txt	hace 16 dias	6.64 kB
<input type="checkbox"/>	OUT82_TF_DNA71.txt	hace 16 dias	11.9 kB
<input type="checkbox"/>	OUT83_TF_DNA71.txt	hace 11 dias	53.4 kB
<input type="checkbox"/>	OUT84_TF_DNA71.txt	hace 16 dias	22.5 kB
<input type="checkbox"/>	OUT86_TF_DNA71.txt	hace 16 dias	27.8 kB
<input type="checkbox"/>	OUT87_TF_DNA71.txt	hace 16 dias	33.1 kB
<input type="checkbox"/>	OUT88_TF_DNA71.txt	hace 15 dias	53.4 kB
<input type="checkbox"/>	Out_Combinatorial_datareal.txt	hace 8 meses	370 kB
<input type="checkbox"/>	OUT_TCS.txt	hace 2 años	1.95 MB

Figure 14: You can visualize this file in your environment path on Jupyter platform.



```

In [1]: # File path (change this if needed)
file_path = "OUT71_TF_DNASage.txt"

# File path - update this to your file location

# Safe Local namespace to store matrices
namespace = {}

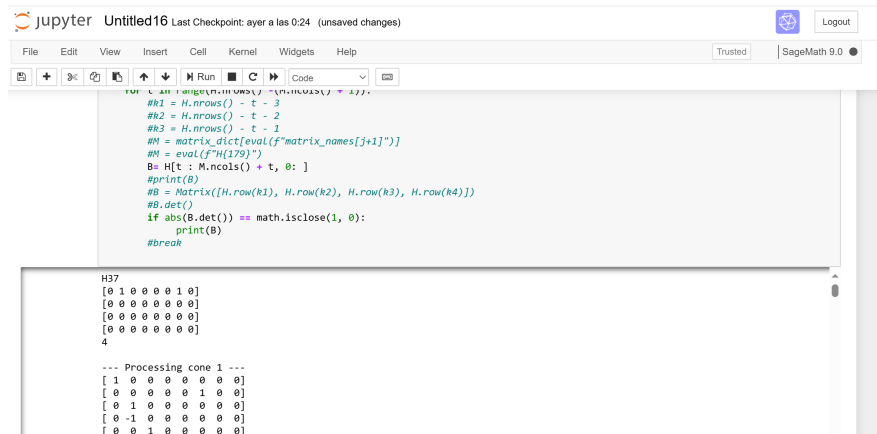
# Use Sage globals so 'Matrix', 'ZZ' are defined
with open(file_path, "r") as f:
    for line in f:
        line = line.strip()
        if line:
            exec(line, globals(), namespace) # - FIXED: pass Sage Globals here

# Collect matrices, sorted by index (e.g. H1, H2, ..., H28)
L = [namespace[k] for k in sorted(namespace, key=lambda x: int(x[1:]))]

matrix_dict = {key: namespace[key] for key in sorted(namespace) if key.startswith("H")}
matrix_names = list(matrix_dict.keys())
print(eval(f"matrix_names[300]"))
print(matrix_dict[eval(f"matrix_names[277]")])
N = matrix_dict[eval(f"matrix_names[300]")]
print(N.nrows())
# Example usage
#print(f"Loaded {len(L)} matrices.")

```

Figure 15: Now, to execute the script for monoid on Sagemath platform. Write correctly the name of you out file, in this case: OUT71_TF_DNASage.txt



```

...
H37
[0 1 0 0 0 1 0]
[0 0 0 0 0 0 0]
[0 0 0 0 0 0 0]
[0 0 0 0 0 0 0]
4
--- Processing cone 1 ---
[ 1 0 0 0 0 0 0]
[ 0 0 0 0 0 1 0]
[ 0 1 0 0 0 0 0]
[ 0 -1 0 0 0 0 0]
[ 0 0 1 0 0 0 0]

```

Figure 16: Run the script and the processing of the first matrices will be visualized on your screen.

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \text{Permutation of rows, } \sim \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \text{Permutation of rows, } \sim \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \text{Permutation of rows, } \sim \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

By linear transformation such as permutation in rows and transpositions of columns and rows. It produces the first matrix in the fourth column (Table 1). Finally, we sum the permuted matrices, therefore, **the total Gene Matrix interaction is:**

$$M(\text{Total}) = \begin{pmatrix} 3 & 1 & 1 & 1 & 0 & 0 & 1 & 1* \\ 1 & 4 & 3 & 0 & 0 & 0 & 0 & 0* \\ 0 & 3 & 4 & 0 & 1 & 0 & 0 & 0 \\ 2 & 0 & 0 & 5 & 0 & 1 & 0 & 0* \\ 0 & 0 & 0 & 1 & 6 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 6 & 0 & 0* \\ 1 & 0 & 0 & 0 & 0 & 0 & 7 & 0* \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 7* \end{pmatrix}$$

Multiply the lattice vector $(1, 0, 0, 0, 0, 0, -g, 1)$, only by the marked rows in asterisk. **The Genotype Arithmetic for case n=7 and in agreement formula (3), is given by:**

$$3 - g + 1 \geq 0,$$

$$1 > 0,$$

$$2 > 0,$$

$$1 > 0,$$

$$1 - 7g \geq 0,$$

$$7 > 0.$$

We summarize all these inequalities. $16 - 8g \geq 0$, implies. $\mathbf{g} = \mathbf{2.0}$.

2.1 Example for length of the pathway, case n=3.

You only need to write in new terminal on Jupyter, the following instruction, Fig 18.

Once chosen, the cones 26 and 27. You can see that the yellow rows conform to the identity matrix with size 4×4 . As is shown below:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \text{ Permuting rows, 1 with 2 } \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \text{ Permuting rows, 1-2 and 2-3 } \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \text{ Permuting rows, 1-2, 2-3, 3-4 } \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \text{ Permuting rows, 1 with 2 } \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The total sum of these matrices is given by the total gene matrix interaction:

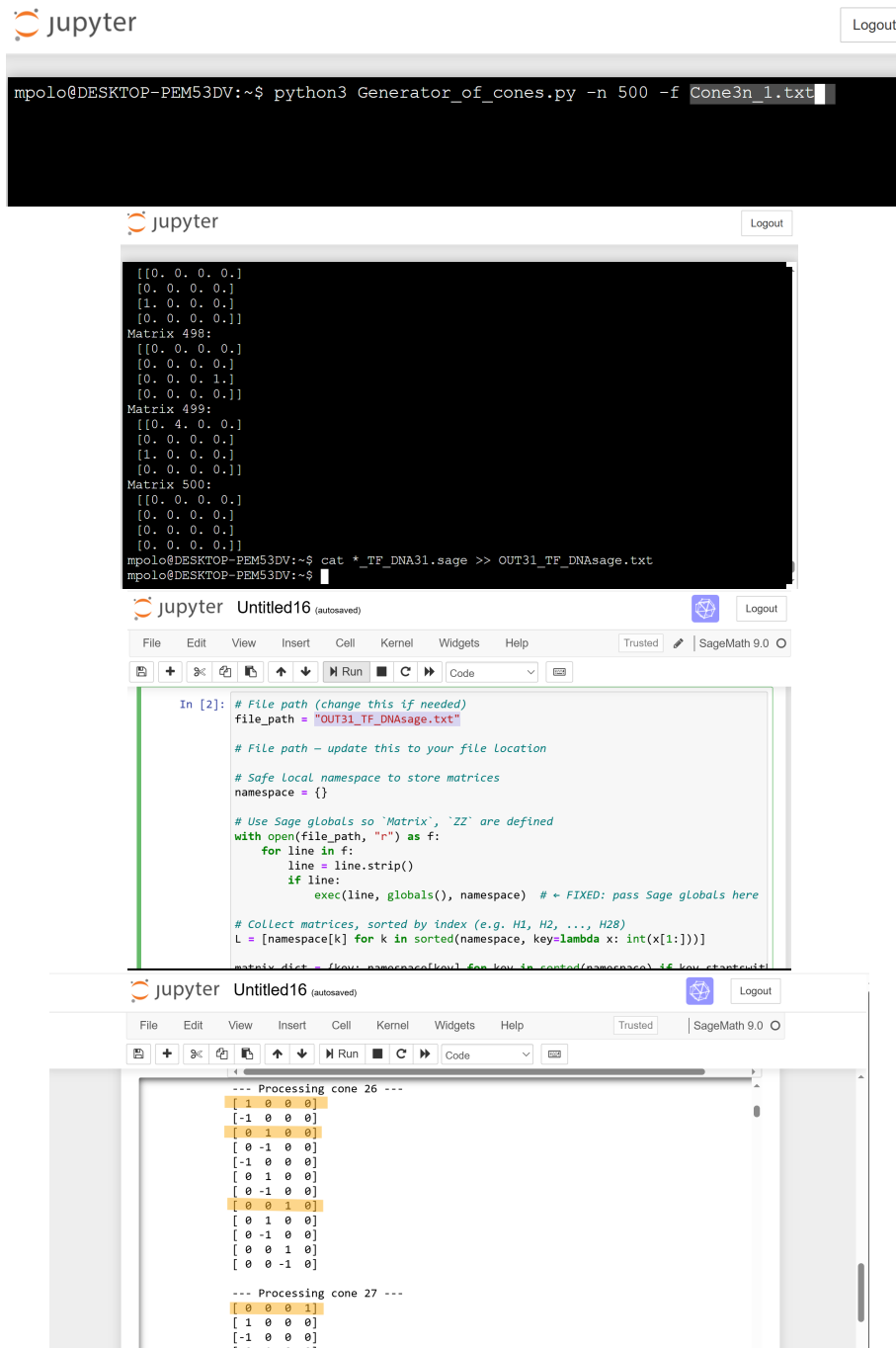


Figure 18: Generation of random orthogonal case for file name: Cone3n_1.txt. Generation of output file with name: OUT31.TF_DNAsage.txt. We write the file name, OUT31.TF_DNAsage.txt in the top of script: Sagemathscript.txt and Run this script. Finally it will be visible 500 generated random matrices. We chose the cones 26 and 27.

$$M(\text{Total}) = \begin{pmatrix} 2 & 2 & 0 & 0* \\ 1 & 0 & 3 & 0 \\ 1 & 1 & 1 & 1* \\ 1 & 0 & 0 & 3* \end{pmatrix}$$

We multiply the informative lattice vector $(1, 0, -g, 1)$ row by row, only by the rows in asterisk, and hence.

The Genotype Arithmetic is given by:

$$(2, 2, 0, 0) * (1, 0, -g, 1) = 2 \geq 0,$$

$$(1, 1, 1, 1) * (1, 0, -g, 1) = 1 - g + 1 \geq 0,$$

$$(1, 0, 0, 3) * (1, 0, -g, 1) = 1 + 3 \geq 0,$$

Add all the inequalities. It yields: $8 - g \geq 0$, $g \leq 8$. Therefore, we take $g = 8$.

References

- [1] Cox D, Little J, O'Shea D. Ideals, varieties, and algorithms: an introduction to computational geometry and commutative algebra. 2nd ed. Springer; 2004.
- [2] Brualdi RA. Combinatorial matrix classes. Cambridge University Press; 2010. Available from: <https://doi.org/10.1017/CBO9780511721182>.
- [3] Kluyver, T., Ragan-Kelley, B., Fernando P'erez, Granger, B., Bussonnier, M., Frederic, J., ... Willing, C. (2016). Jupyter Notebooks – a publishing format for reproducible computational workflows. In F. Loizides B. Schmidt (Eds.), Positioning and Power in Academic Publishing: Players, Agents and Agendas (pp. 87–90).
- [4] William Stein and David Joyner. SAGE: *System for Algebra and Geometry Experimentation*. ACM SIGSAM Bulletin, volume 39, number 2, pages 61–64, 2005.