

1 Tutorial to runnig Monoids on Jupyter-notebook and Sagemath: End-Product Inhibition cases.

First you need to install a jupyter notebook platform.

Step-by-step (Windows / macOS / Linux):

Go to: <https://www.anaconda.com/products/distribution>

Download the installer for your OS:

-Windows: .exe

-macOS: .pkg

-Linux: .sh

Run the installer and follow the instructions (default settings are fine).

After installation:

Open Anaconda Navigator and click on "Launch" under Jupyter Notebook,
OR

Open a terminal / Anaconda Prompt and run: `jupyter notebook`, in Ubuntu:
`jupyter-notebook`.

Once installed your jupyter-notebook platform. You follow the screenshots to runnig monoids for end-product inhibition cases.



Figure 1: Open a new notebook for Python 3.

The screenshot shows a Jupyter Notebook titled 'Untitled15' with a status bar indicating 'Last Checkpoint: hace 2 minutos (unsaved changes)'. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running, and code execution. The notebook contains three code cells:

```

In [ ]: # Update and install
        !apt-get update -qq
        !apt-get install -y -qq sagemath

In [ ]: # versión
        !sage --version

In [ ]: # Run Sagemath example
        sage -c 'R.<x,y> = QQ[]; I = ideal(x^2+y^2-1); I.groebner_basis()'

```

Figure 2: The next step is to install the platform to run algorithms in computational algebraic geometry. In our case to compute generators in monoids, you only copy and paste or simply write the code lines shown in order as is shown in figure.

Note: The script only run monoids for cones of dimension $n = 1$ up to $n = 9$. The corecto name files for each case of length of the pathways $n = 1, 2, \dots, 9$, are given in the following: Cone3n_1.txt, Cone3n_2.txt, Cone3n_3.txt, Cone3n_4.txt, Cone4n_1.txt, Cone4n_2.txt, Cone4n_3.txt, Cone4n_4.txt, Cone4n_5.txt, Cone5n_1.txt, Cone5n_2.txt, Cone5n_3.txt, Cone5n_4.txt, Cone5n_5.txt, Cone5n_6.txt, Cone6n_1.txt, Cone6n_2.txt, Cone6n_3.txt, Cone6n_4.txt, Cone6n_5.txt, Cone6n_6.txt, Cone6n_7.txt, Cone7n_1.txt, Cone7n_2.txt, Cone7n_3.txt, Cone7n_4.txt, Cone7n_5.txt, Cone7n_6.txt, Cone7n_7.txt, Cone7n_8.txt, Cone8n_1.txt, Cone8n_2.txt, Cone8n_3.txt, Cone8n_4.txt, Cone8n_5.txt, Cone8n_6.txt, Cone8n_7.txt, Cone8n_8.txt, Cone8n_9.txt, Cone9n_1.txt, Cone9n_2.txt, Cone9n_3.txt, Cone9n_4.txt, Cone9n_5.txt, Cone9n_6.txt, Cone9n_7.txt, Cone9n_8.txt, Cone9n_9.txt, Cone9n_10.txt.

Note: In this linux command : `cat *_TF_DNA31.sage >> OUT31_TF_DNAsage.txt`. You only must to change the number in the name file. If you are in the case for length of the pathway $n = 4$. The output files, are written as:

```

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

```

```

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: In the following, open the python script file called: **Generator of cones.py**. It which is uploaded in tree of gitgub. Copy and paste in a new notebook in python.



Figure 4: Once installed Sagemath version 9.0- Open a notebook in Sagemath.

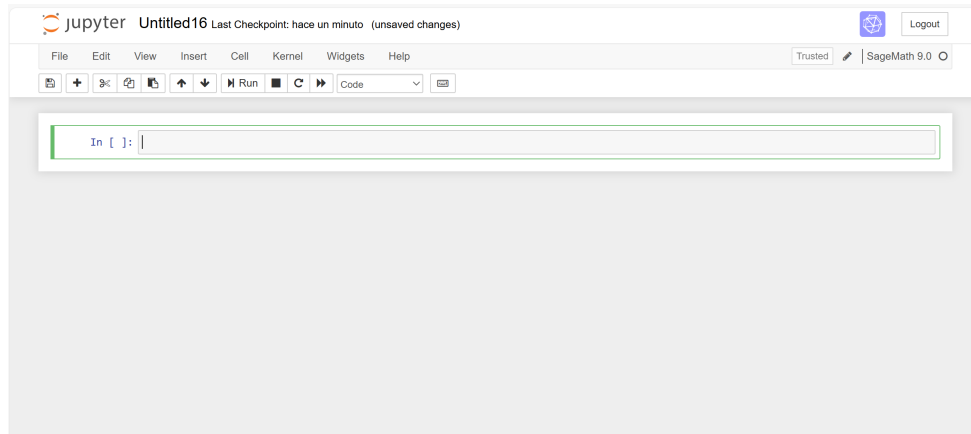


Figure 5: It will open a newbook in Sagemath. In a moment you will run a script in this prompt. You leave it open.

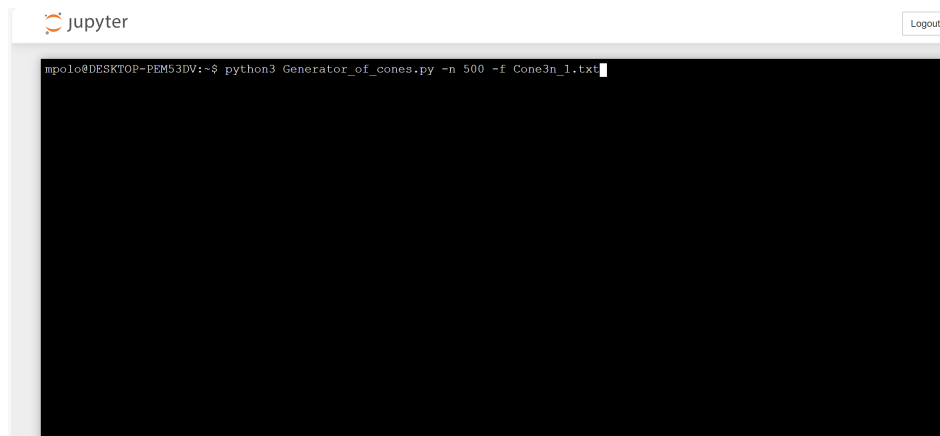


Figure 6: Open a new terminal in Jupyter. You write in the prompt the line shown in the figure. Only follow and respect the name files: Cone3n_1.txt, Cone3n_2.txt,..., or Cone7n_1.txt, Cone7n_2.txt, Cone7n_3,...,Cone7n_8.txt and similar for simulation of more cones and monoids for different length of pathways n . The instruction $-n$ 500 lets to simulate 500 random matrices through Boltzmann distribution.

```

jupyter
Logout

[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_DNAsage.txt

```

Figure 7: After you have generated 500 random matrices. You can write the instruction shown on the screenshot with the objective to save the whole random matrices in an one file called OUT31_TF_DNAsage.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_DNAsage.txt	hace 17 días	28.4 kB
<input checked="" type="checkbox"/>	OUT31_TF_DNAsage.txt	hace unos segundos	36.9 kB
<input type="checkbox"/>	OUT41_TF_DNAsage.txt	hace 10 días	42.9 kB
<input type="checkbox"/>	OUT42_TF_DNAsage.txt	hace 10 días	35.4 kB
<input type="checkbox"/>	OUT44_TF_DNAsage.txt	hace 9 días	35.4 kB
<input type="checkbox"/>	OUT61_TF_DNAsage.txt	hace 13 días	54.9 kB
<input type="checkbox"/>	OUT62_TF_DNAsage.txt	hace 10 días	44.4 kB
<input type="checkbox"/>	OUT720_TF_DNAsage.txt	hace 14 días	10.1 kB

Figure 8: In the following you can verify the content of this file in the Jupyter tree as is shown in the image.

```

471 H73 = Matrix(ZZ, 4, 4, [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, 1, 0, 0, 1, 0, 0, 0, 0])
473 H75 = Matrix(ZZ, 4, 4, [0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 1, 0, 0, 0, 0])
474 H76 = Matrix(ZZ, 4, 4, [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0])
475 H77 = Matrix(ZZ, 4, 4, [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0])
476 H78 = Matrix(ZZ, 4, 4, [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 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])
478 H7 = Matrix(ZZ, 4, 4, [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0])
479 H80 = Matrix(ZZ, 4, 4, [0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0])
480 H81 = Matrix(ZZ, 4, 4, [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 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])
482 H83 = Matrix(ZZ, 4, 4, [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 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])
484 H85 = Matrix(ZZ, 4, 4, [0, 0, 0, 0, 0, 0, 0, 0, 1, 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])
486 H87 = Matrix(ZZ, 4, 4, [0, 0, 0, 0, 0, 0, 0, 0, 1, 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])
488 H89 = Matrix(ZZ, 4, 4, [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0])
489 H8 = Matrix(ZZ, 4, 4, [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0])
490 H90 = Matrix(ZZ, 4, 4, [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])
492 H92 = Matrix(ZZ, 4, 4, [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0])
493 H93 = Matrix(ZZ, 4, 4, [0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 2, 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])
495 H95 = Matrix(ZZ, 4, 4, [0, 0, 0, 1, 0, 0, 0, 0, 1, 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])

```

Figure 9: The content of the file it looks as is shown in the image. This files are prepared in format for Sagemath platform.

```

In [ ]: # File path (change this if needed)
file_path = "OUT31_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

```

Figure 10: Now you come back to the notebook that you have left open in figure 5. Copy and paste the script called Sagemathscript.txt, this file text is uploaded in the github link. You must write in the begining of this script the right name of the file, in this case was "OUT31_TF_DNAsage.txt".

The screenshot shows a Jupyter Notebook window titled "Untitled16" with a "Last Checkpoint: hace 21 minutos (unsaved changes)" status. The interface includes a top menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running, and saving. The code editor contains the following script:

```
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]
```

The output area at the bottom shows the execution of the code, with the matrix representations for cones 441, 442, and 443 displayed. The status bar at the bottom indicates "In 1.1s".

Figure 11: Once you have copied and pasted the script in the Sagemathscrip.txt file. You can run it. In the following you will 500 matrix representation for monoids for the length of the pathway n that you have chosen.