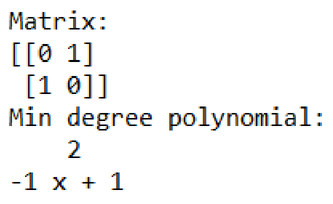
1 Methodology

Our algorithm, contained in min_degree.py, first converts each matrix power of the given $(n \times n)$ matrix m into individual column vectors up to the matrix power n. The result is an $((n^2) \times (n+1))$ matrix M, whose columns are the matrix powers of the m. However, we do not always go to $n+1$ columns. After adding each new column to M, we check if M has a non-zero solution for the null_space. If this exists, we interrupt our loop early, as we have found the minimal polynomial of m.

2 Tests

First, we test a simple matrix: $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Our program outputs the following:



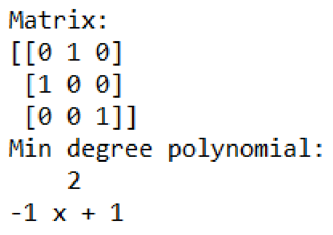
```
Matrix:
[[0 1]
 [1 0]]
Min degree polynomial:
    2
-1 x + 1
```

We can check by adding the negative 2nd matrix power to the identity matrix:

$$-\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

When substituting the variable x with our matrix power, we see that the solution is 0. Therefore, this is an annihilating polynomial for our matrix. However, we must make sure that this is the minimal polynomial.

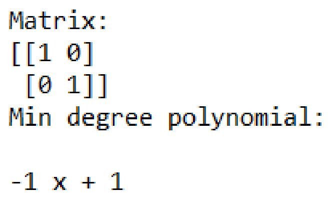
If we extend the previous matrix to be: $\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$, we know that the same solution as above applies (since it's matrix powers are also the identity matrix), but there may also exist a solution as high as degree 3. When we try this matrix in our program, we get:



```
Matrix:
[[0 1 0]
 [1 0 0]
 [0 0 1]]
Min degree polynomial:
    2
-1 x + 1
```

Here we see that a solution of degree 2 was found. The program did not return the solution that also exists at degree 3. This shows that our program is correctly returning once it finds the **minimum** degree solution that annihilates the matrix.

Let's try the identity matrix: $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

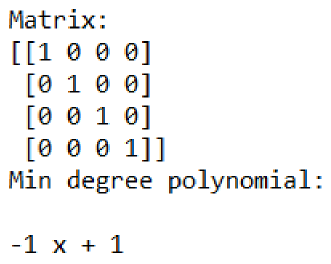


```
Matrix:
[[1 0]
 [0 1]]
Min degree polynomial:

-1 x + 1
```

We can easily verify the solution: $-\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$
This is another case where a solution also exists as a polynomial of degree 2, but we returned the **minimum** degree polynomial that annihilates the matrix. A second degree solution could be $x^2 + x - 2$.

We know that the identity matrix of size 4 should have the same solution:

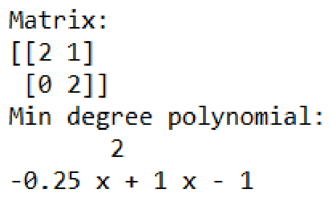


```
Matrix:
[[1 0 0 0]
 [0 1 0 0]
 [0 0 1 0]
 [0 0 0 1]]
Min degree polynomial:

-1 x + 1
```

Our program output again displays the minimum polynomial for this case.

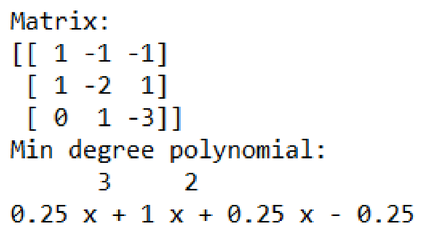
Here is the output for a simple jordan block:



```
Matrix:
[[2 1]
 [0 2]]
Min degree polynomial:
       2
-0.25 x + 1 x - 1
```

Verify the solution: $-.25 \begin{bmatrix} 4 & 4 \\ 0 & 4 \end{bmatrix} + \begin{bmatrix} 2 & 1 \\ 0 & 2 \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$

Finally, we test an example matrix from the wikipedia article about minimal polynomials: `https://en.wikipedia.org/wiki/Minimal_polynomial_(linear_algebra)`.



```
Matrix:
[[ 1 -1 -1]
 [ 1 -2  1]
 [ 0  1 -3]]
Min degree polynomial:
      3     2
0.25 x + 1 x + 0.25 x - 0.25
```

The article indicates the solution should be $x^3 + 4x^2 + x - I$. Their solution is a multiple of ours, therefore our solution is correct.

Within main.py there are several more examples. By executing main.py, you can verify that my program produces all output already shown, and also includes several more test cases that are similar to the ones already discussed in this write-up.