14_P3_Assessment_GionRubitschung

October 2, 2023

Compute the rref of

$$M = \begin{pmatrix} 0 & 0 & 1 & 2 & 1 & 1 & 0 \\ -2 & -4 & 2 & 4 & 4 & 0 & 0 \\ 1 & 2 & 1 & 2 & 0 & 1 & 1 \\ 2 & 4 & 1 & 2 & -1 & -2 & 5 \\ 2 & 4 & -1 & -2 & -3 & -2 & 3 \end{pmatrix}$$

and show every step of the computation

[1]:
```
# Needed for lib import, since it is a local module
import sys

sys.path.insert(0, "..")

from lib.matrix_operations import add_row, swap_row, multiply_row
import numpy as np
```

[2]:
```
M = np.array(
    [
        [0, 0, 1, 2, 1, 1, 0],
        [-2, -4, 2, 4, 4, 0, 0],
        [1, 2, 1, 2, 0, 1, 1],
        [2, 4, 1, 2, -1, -2, 5],
        [2, 4, -1, -2, -3, -2, 3],
    ]
)
print(M)
```

```
[[ 0  0  1  2  1  1  0]
 [-2 -4  2  4  4  0  0]
 [ 1  2  1  2  0  1  1]
 [ 2  4  1  2 -1 -2  5]
 [ 2  4 -1 -2 -3 -2  3]]
```

1 Clean C1

1. Swap $R_1 \leftrightarrow R_2$

[3]:
```
M = swap_row(M, 2, 0)
print(M)
```

```
[[ 1  2  1  2  0  1  1]
 [-2 -4  2  4  4  0  0]
 [ 0  0  1  2  1  1  0]
 [ 2  4  1  2 -1 -2  5]
 [ 2  4 -1 -2 -3 -2  3]]
```

$M[0,0] = 1 \rightarrow pivot$

2. $R_1 = R_1 + 2R_0$
3. $R_3 = R_3 - 2R_0$
4. $R_4 = R_4 - 2R_0$

[4]:
```
M = add_row(M, row_one=1, row_two=0, factor=2)
M = add_row(M, row_one=3, row_two=0, factor=-2)
M = add_row(M, row_one=4, row_two=0, factor=-2)
print(M)
```

```
[[ 1  2  1  2  0  1  1]
 [ 0  0  4  8  4  2  2]
 [ 0  0  1  2  1  1  0]
 [ 0  0 -1 -2 -1 -4  3]
 [ 0  0 -3 -6 -3 -4  1]]
```

2 Clean C2

1. Swap $R_2 \leftrightarrow R_1$

[5]:
```
M = swap_row(M, row_one=2, row_two=1)
print(M)
```

```
[[ 1  2  1  2  0  1  1]
 [ 0  0  1  2  1  1  0]
 [ 0  0  4  8  4  2  2]
 [ 0  0 -1 -2 -1 -4  3]
 [ 0  0 -3 -6 -3 -4  1]]
```

$M[1,2] = 1 \rightarrow pivot$

2. $R_0 = R_0 - R_1$
3. $R_2 = R_2 - 4R_1$
4. $R_3 = R_3 + R_1$
5. $R_4 = R_4 + 3R_1$

[6]:
```
M = add_row(M, row_one=0, row_two=1, factor=-1)
M = add_row(M, row_one=2, row_two=1, factor=-4)
M = add_row(M, row_one=3, row_two=1, factor=1)
M = add_row(M, row_one=4, row_two=1, factor=3)
```

```
print(M)
```

```
[[ 1  2  0  0 -1  0  1]
 [ 0  0  1  2  1  1  0]
 [ 0  0  0  0  0 -2  2]
 [ 0  0  0  0  0 -3  3]
 [ 0  0  0  0  0 -1  1]]
```

3 Clean C3

1. Swap $R_2 \leftrightarrow R_4$
2. $R_2 \leftarrow -R_2$

[7]:
```
M = swap_row(M, row_one=2, row_two=4)
M = multiply_row(M, row=2, factor=-1)
print(M)
```

```
[[ 1  2  0  0 -1  0  1]
 [ 0  0  1  2  1  1  0]
 [ 0  0  0  0  0  1 -1]
 [ 0  0  0  0  0 -3  3]
 [ 0  0  0  0  0 -2  2]]
```

$M[2,5] = 1 \rightarrow pivot$

3. $R_1 = R_1 - R_2$
4. $R_3 = R_3 + 3R_2$
5. $R_4 = R_4 + 2R_2$

[8]:
```
M = add_row(M, row_one=1, row_two=2, factor=-1)
M = add_row(M, row_one=3, row_two=2, factor=3)
M = add_row(M, row_one=4, row_two=2, factor=2)
print(M)
```

```
[[ 1  2  0  0 -1  0  1]
 [ 0  0  1  2  1  0  1]
 [ 0  0  0  0  0  1 -1]
 [ 0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0]]
```

4 Solution

$$rref(M) = \begin{pmatrix} 1 & 2 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 1 & 2 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

5 Appendix

5.1 lib module

5.1.1 matrix_operations

```
def add_row(
    X: np.ndarray, row_one: int, row_two: int, factor: int, matrix_type: str = "int"
) -> np.ndarray:
    """Operates an addition of two rows inside a given matrix `X`

    Args:
        X (np.ndarray): The given matrix
        row_one (int): Index of row one in `X`
        row_two (int): Index of row two in `X`
        factor (int): Factor applied to `X[row_two]` how much to the row `X[row_one]` adds,
            use negative values for substraction
        matrix_type (str, optional): Type of the matrix entries,
            for example `float64` or `int`. Defaults to `int`.

    Raises:
        InvalidSpanOperationException: If `row_one` equals `row_two`,
            the operation cannot be done

    Returns:
        np.ndarray: The edited matrix
    """
    if row_one == row_two:
        raise InvalidSpanOperationException()

    X = X.astype("float64")
    X[row_one] += factor * X[row_two]
    return X.astype(matrix_type)


def multiply_row(
    X: np.ndarray, row: int, factor: int, matrix_type: str = "int"
) -> np.ndarray:
    """Operates a multiplication of a row inside a given matrix `X`

    Args:
        X (np.ndarray): The given matrix
        row (int): Index of the row to multiply in `X`
        factor (int): Factor of how much to multiply
        matrix_type (str, optional): Type of the matrix entries,
            for example `float64` or `int`. Defaults to `int`.

    Raises:
        InvalidSpanOperationException: If `factor` equals `0`,
```

```
            the multiplication is invalid

    Returns:
        np.ndarray: The edited matrix
    """
    if factor == 0:
        raise InvalidSpanOperationException()
    X = X.astype("float64")
    X[row] = factor * X[row]
    return X.astype(matrix_type)


def swap_row(
    X: np.ndarray, row_one: int, row_two: int, matrix_type: str = "int"
) -> np.ndarray:
    """Operates a swap of row inside a given matrix `X`

    Args:
        X (np.ndarray): The given matrix
        row_one (int): Index of row 1
        row_two (int): Index of row 2
        matrix_type (str, optional): Type of the matrix entries,
            for example `float64` or `int`. Defaults to `int`.

    Returns:
        np.ndarray: The edited matrix
    """
    X = X.astype("float64")
    X[[row_one, row_two]] = X[[row_two, row_one]]
    return X.astype(matrix_type)
```

5.1.2 errors

```
class InvalidSpanOperationException(Exception):
    """Exception raised when when a matrix operation would change the span"""

    def __init__(self) -> Exception:
        super().__init__("Not allowed, this could change the span!")
```