CAD/CAE

Zadanie 2

Autor: Jan Augustyn

1. Modyfikacja kodu źródłowego

```
% Code modifications

coeff_nrows = size(coeffs, 1);
coeff_ncols = size(coeffs, 2);
if (coeff_nrows ~= nry || coeff_ncols ~= nrx)
    disp("[Input Error]: It's definitely one of the coeffs of all time:")
    fprintf("\tGot: nrows: %d, ncolumns: %d\n", coeff_nrows, coeff_ncols)
    fprintf("\tExpected: nrows: %d, ncolumns: %d\n\n", nry, nrx)
    return
end

M = zeros(size(X));
for i=1:nrx
  vx=compute_spline(knot_vectorx,px,i,X);
  for j=1:nry
    vy=compute_spline(knot_vectory,py,j,Y);
    M = M + coeffs(j, i) * vx .* vy;
  end
end

figure('Name','Maze 3D view');
surf(X,Y, M);
set(gca, "YDir", "reverse");

figure('Name','Maze 3D view cropped');
surf(X,Y, M);
zlim([0.7, 1])
set(gca, "YDir", "reverse");

% Code modifications end
```

Kod 1 - Modyfikacja kodu funkcji 'spline2Duniform' rysująca trójwymiarowy wykres labiryntu.

2. Wczytywanie danych wejściowych z pliku

```
function spline2D_comp()
data = load('data/21x25.mat');
spline2Duniform(data.knot_vectorx, data.knot_vectory, data.coeffs);
return
```

Kod 2 - Kod funkcji spline2D_comp pozwalający na wczytanie danych wejściowych z pliku '.mat'.

3. Kod skryptu generującego macierz i zapisującego dane wejściowe

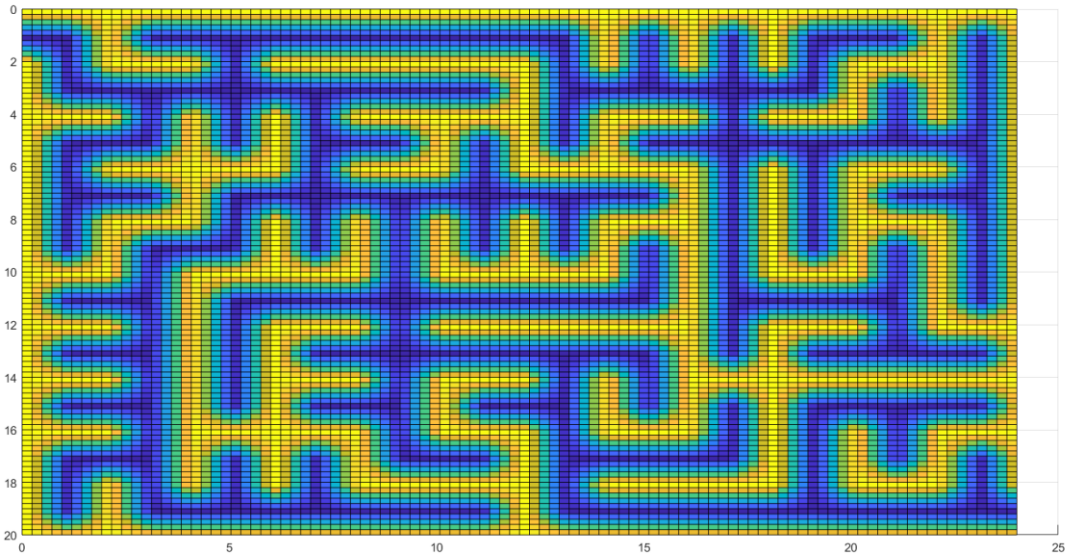
```python
from mazelib import Maze
from mazelib.generate.Prims import Prims
import numpy as np
import scipy.io
import sys

def get_knot_vectors(dimension: int, lenght: int):
    beg = [0 for _ in range(dimension)]
    mid = [i for i in range(1, lenght-1)]
    end = [lenght-1 for _ in range(dimension)]
    return beg+mid+end

def add_outer_entrances(grid: list[list[int]]):
    grid[1][0] = 0
    grid[-2][-1] = 0

def main():
    if len(sys.argv) not in (3, 4):
        print(f"[Usage]: python {sys.argv[0]} [row halls] [col halls] [optional: file name]")
        return

    row_halls = int(sys.argv[1])
    col_halls = int(sys.argv[2])

    file_name = "data"
    if len(sys.argv) == 4:
        file_name = sys.argv[3]

    m = Maze()
    m.generator = Prims(row_halls, col_halls)
    m.generate()

    add_outer_entrances(m.grid)

    vectorx_len = 2 * col_halls + 1
    vectory_len = 2 * row_halls + 1
    knot_vectorx = get_knot_vectors(2, vectorx_len)
    knot_vectory = get_knot_vectors(2, vectory_len)

    with open(file_name + ".txt", "w") as file:
        file.write(f"knot_vectorx = {knot_vectorx};\n\n")
        file.write(f"knot_vectory = {knot_vectory};\n\n")
        file.write(f"coeffs = {np.array2string(np.array(m.grid), separator=' ', threshold=10**10, max_line_width=10*10)};\n")

    knot_vectorx = np.array(knot_vectorx, dtype=np.float64)
    knot_vectory = np.array(knot_vectory, dtype=np.float64)
    coeffs = np.array(m.grid, dtype=np.float64)

    scipy.io.savemat(f'{file_name}.mat', {
        'knot_vectorx': knot_vectorx,
        'knot_vectory': knot_vectory,
        'coeffs': coeffs})

    print(f"Maze data successfuly saved to {file_name}.txt and {file_name}.mat")

if __name__=="__main__":
    main()
```

Kod 3 - Skrypt generujący losową macierz o zadanym rozmiarze i zapisujący dane wejściowe w plikach '.mat' i '.txt'.

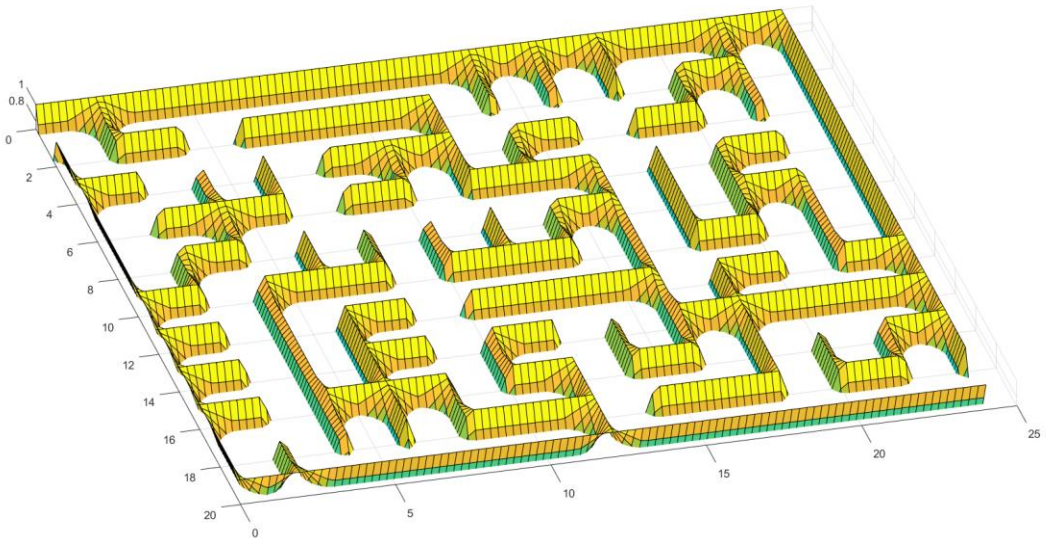
4. Wygenerowane dane wejściowe dla labiryntu o rozmiarze 21x25

```
knot_vectorx = [0, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 24];


knot_vectory = [0, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 20];


coeffs = [
 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
 [0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 1 0 1]
 [1 0 1 1 1 0 1 1 1 1 1 1 1 0 1 0 1 0 1 0 1 1 1 0 1]
 [1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 1 0 1]
 [1 1 1 0 1 0 1 0 1 1 1 1 1 0 1 1 1 0 1 1 1 0 1 0 1]
 [1 0 0 0 1 0 1 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 1]
 [1 0 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 0 1 0 1 1 1 0 1]
 [1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 1]
 [1 0 1 1 1 0 1 0 1 0 1 0 1 0 1 1 1 0 1 0 1 1 1 0 1]
 [1 0 1 0 0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1]
 [1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 0 1 1 1 0 1 0 1]
 [1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 1]
 [1 1 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1]
 [1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1]
 [1 1 1 0 1 0 1 1 1 0 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1]
 [1 0 0 0 1 0 1 0 0 0 1 0 1 1 1 0 1 0 1 0 1 0 0 0 0 1]
 [1 1 1 0 1 1 1 1 1 0 1 1 1 0 1 1 1 0 1 0 1 0 1 1 1]
 [1 0 0 0 1 0 1 0 1 0 0 0 1 0 0 0 0 0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 0 1]
 [1 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0]
 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
];
```

1. Wygenerowany wykres labiryntu o rozmiarze 21x25

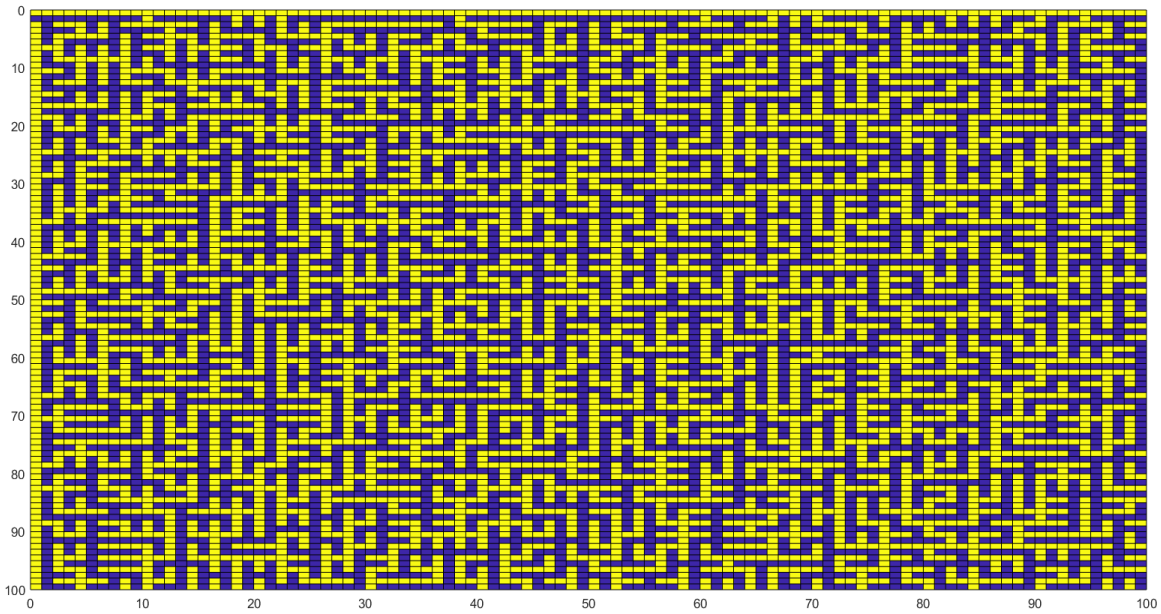


Wykres 1 - Trójwymiarowy wykres otrzymanego labiryntu o rozmiarze 21x25 z widokiem od góry.

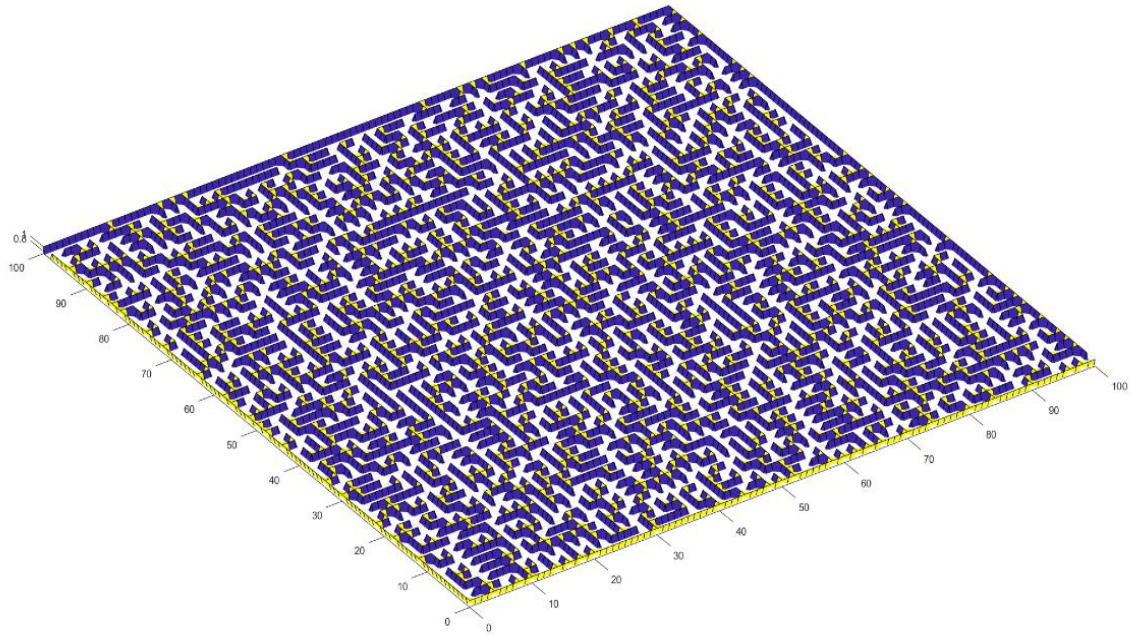


Wykres 2 - Trójwymiarowy wykres otrzymanego labiryntu o rozmiarze 21x25 z widokiem od boku.

2. Wygenerowany wykres labiryntu o rozmiarze 101x101

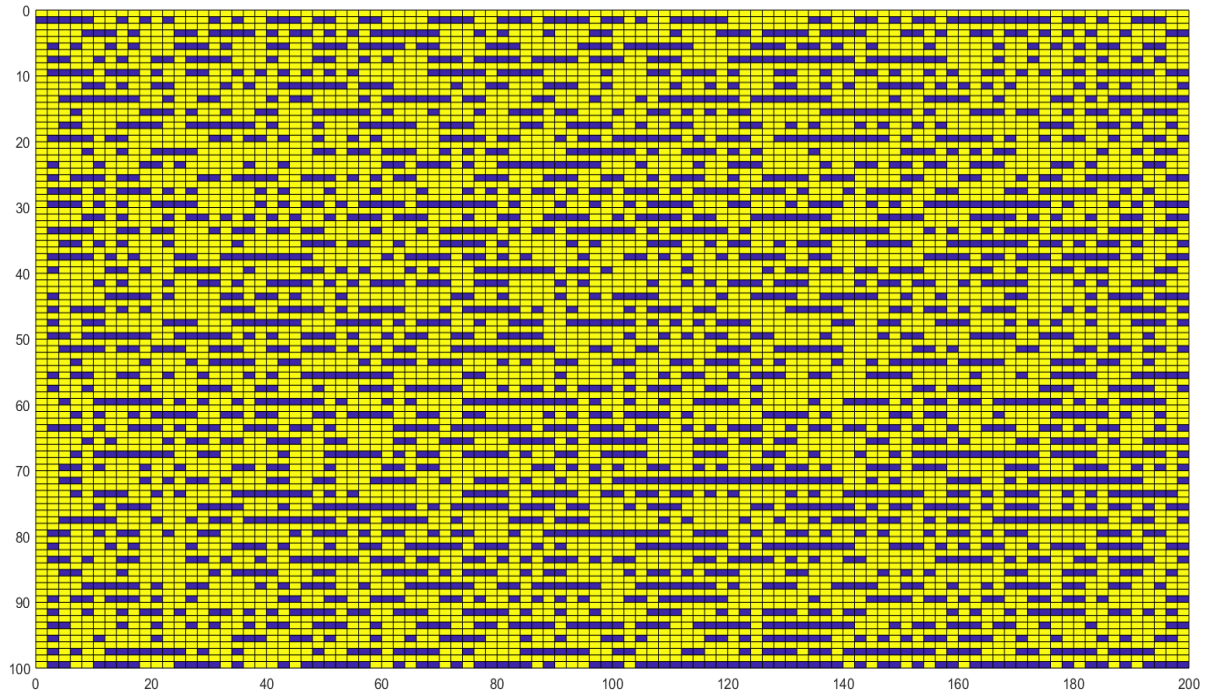


Wykres 3 - Trójwymiarowy wykres otrzymanego labiryntu o rozmiarze 101x101 z widokiem od góry.

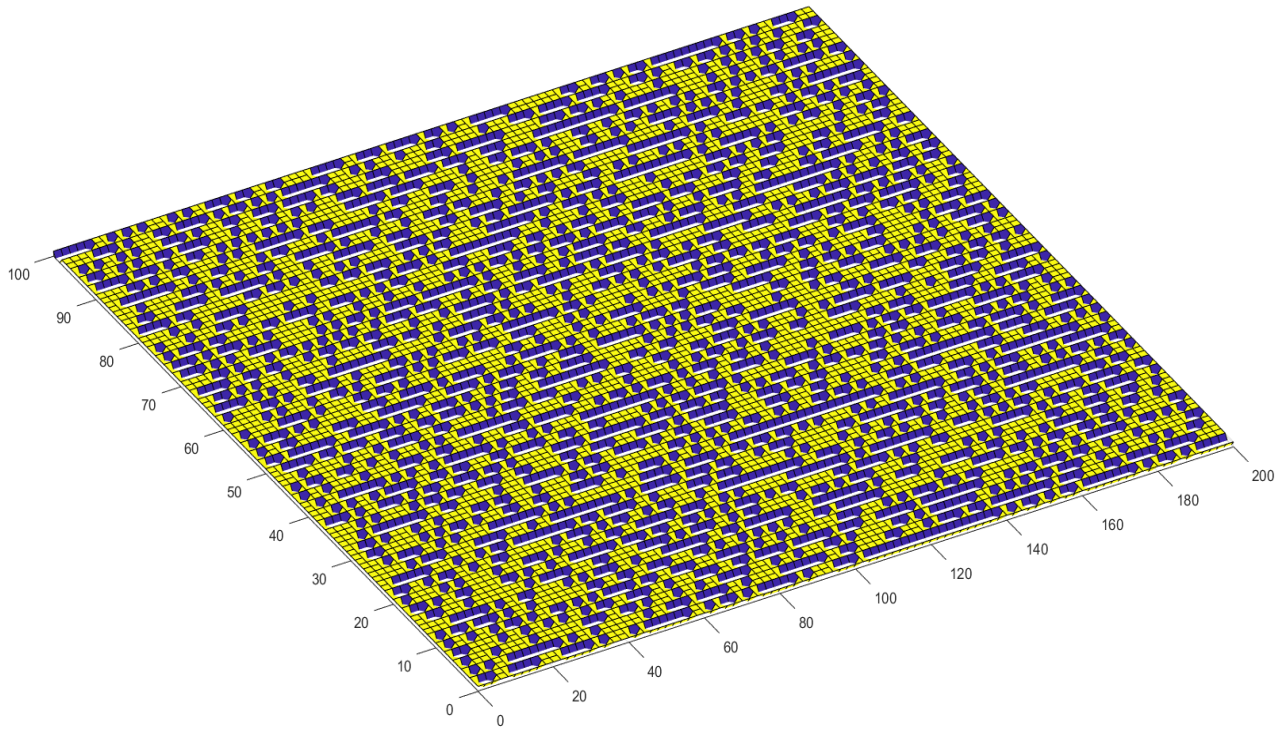


Wykres 4 - Trójwymiarowy wykres otrzymanego labiryntu o rozmiarze 101x101 z widokiem od boku.

3. Wygenerowany wykres labiryntu o rozmiarze 101x201



Wykres 5 - Trójwymiarowy wykres otrzymanego labiryntu o rozmiarze 101x201 z widokiem od góry.



Wykres 6 - Trójwymiarowy wykres otrzymanego labiryntu o rozmiarze 101x201 z widokiem od boku.