

Dopełnienie Schura

December 6, 2021

1 Dopełnienie Schur'a z użyciem eliminacji Gaussa

Maciej Skoczeń, Kacper Kafara

grupa wtorek (A) 17:50

1.1 Środowisko obliczeniowe

Processor: Intel i7-9750H @ 2,6 GHz; 6 rdzeni fizycznych (12 log.)

1.2 Importy & typy

```
[1]: import numpy as np
import matplotlib.pyplot as plt
import os
import re
import subprocess
import matplotlib.pyplot as plt

from timeit import default_timer
from pprint import pprint
from math import sqrt

Array = np.ndarray
```

1.3 Funkcje pomocnicze

```
[2]: class Timer(object):
    def __init__(self):
        self._start_time = None
        self._stop_time = None

    def start(self):
        self._start_time = default_timer()

    def stop(self):
        self._stop_time = default_timer()
```

```

@property
def elapsed(self, val = None):
    if self._stop_time is None or self._start_time is None:
        return None
    elapsed = self._stop_time - self._start_time
    return elapsed

# mock impl
def is_int(value) -> bool:
    as_int = int(value)
    return value == as_int

```

1.3.1 Wczytywanie macierzy

wygenerowanej za pomocą dostarczonego skryptu `mass_matrix`

```

[3]: def input_matrix(octave_matrix, n, m, q=1):
    result = np.zeros((n*q, m*q), dtype=np.double)

    for elem in octave_matrix:
        m = re.match(r"\s*\((\d+),\s*(\d+)\)\s*->\s*(\d+\.\d+)\s*", elem)
        if m is not None:
            x, y, value = m.groups()
        elif len(elem) > 0:
            coord, value = elem.strip().split(' -> ')
            value = float(value)
            x, y = coord.split(',')
            x, y = x[1:], y.strip()[:-1]
        else:
            continue

        for i in range(q):
            for j in range(q):
                result[i*n + int(x) - 1, j*n + int(y) - 1] = float(value)

    return result

```

```

[4]: def load_octave_matrix(filename):
    with open(filename, "r") as file:
        return file.readlines()

```

```

[5]: data_dir = "../..output"

def resolve_path(matrix_type, width, height = None, generate = False):
    if height is None: height = width
    path = f"{data_dir}/{matrix_type}-{width}x{height}.txt"
    if os.path.isfile(path): return path

```

```

else:
    if not generate:
        raise FileNotFoundError(f"Matrix file {path} not found")

    # do generowania macierzy potrzebny jest direnv, ustawiona zmienna
    # środowiskowa:
    # SCRIPT_DIR=<path-to-scripts-dir>
    # albo na sztywno ustawiona ścieżka do skryptu (ale wtedy trzeba
    ↪zmodyfikować)
    # funkcję generate_matrix

    if width != height:
        raise ValueError("Can only generate square matrix")

    generate_matrix(matrix_type, width)

    if os.path.isfile(path): return path
    else:
        print(path)
        raise RuntimeError("Failed to generate matrix")

resolve_matrix = lambda matrix_type, n, m, q = 1: input_matrix(
    load_octave_matrix(resolve_path(matrix_type, n, m)), n, m, q
)

def resolve_matrix(matrix_type, n, m, q = 1, generate = False):
    return input_matrix(
        load_octave_matrix(resolve_path(matrix_type, n, m, generate =
    ↪generate))), n, m, q
    )

def generate_matrix(matrix_type, rank):
    if matrix_type not in {'iga', 'fem'}:
        raise ValueError(f"Invalid matrix type: {matrix_type}")

    if rank < 16 or not is_int(sqrt(rank)):
        raise ValueError(f"Invalid matrix rank: {rank}. Must be >= 16 and
    ↪sqrt(rank) must be of type integer.")

    rank_root = int(sqrt(rank))

    if matrix_type == 'fem':
        for p in range(2, 5):
            double_nxx = rank_root - p + 1
            if double_nxx % 2 == 0 and double_nxx // 2 >= 2:
                nxx = double_nxx // 2

```

```

        pxx = p
        break
    else:
        raise RuntimeError(f"Failed to determine nxx, pxx for rank: {rank}")
else:
    for p in range(2, 5):
        nxx = rank_root - p
        if nxx >= 2:
            pxx = p
            break
    else:
        raise RuntimeError(f"Failed to determine nxx, pxx for rank: {rank}")

cwd = os.getcwd()
scripts_dir = os.getenv('SCRIPTS_DIR')
os.chdir(scripts_dir)
!./generate-matrix.sh cpp {matrix_type} {nxx} {pxx} 0
os.chdir(cwd)

```

1.4 Eliminacja Gaussa

```

[6]: def transform_matrix_gaussian_elim(
    A: Array,
    rows_to_transform: int,
    in_place: bool = False,
    timer: Timer = None
) -> Array:

    if not in_place: A = A.copy()

    if timer is not None:
        timer.start()

    n, _ = A.shape
    for i in range(0, min(n - 2, rows_to_transform)):
        for j in range(i + 1, n):
            factor = A[j, i] / A[i, i]
            A[j, i] = 0
            for k in range(i + 1, n):
                A[j, k] -= factor * A[i, k]

    if timer is not None: timer.stop()
    if not in_place: return A

```

1.5 Dopełnienie Schur'a

```
[7]: def schur_complement(A: Array, complement_degree: int, timer: Timer = None) -> Array:
    transformed = transform_matrix_gaussian_elim(A,
                                                A.shape[0] - complement_degree,
                                                in_place = False,
                                                timer = timer)
    return transformed[A.shape[0] - complement_degree :, A.shape[1] - complement_degree :]
```

```
[ ]: nxxs = {}
nxxs['iga'] = [i for i in range(2, 31)]
nxxs['fem'] = [i for i in range(2, 17)]

pxx = 2
rxx = 0
ranks = {}
ranks['iga'] = [(nxx + pxx) ** 2 for nxx in nxxs['iga']]
ranks['fem'] = [(2 * nxx + pxx - 1) ** 2 for nxx in nxxs['fem']]

matrixtypes = 'iga', 'fem'

main_timer = Timer()
exec_times = {
    'iga': {},
    'fem': {}
}
exec_ranks = {
    'iga': {},
    'fem': {},
}

padding = lambda n: n * ' '

for matrix_t in matrixtypes:
    matrices = ((resolve_matrix(matrix_t, rank, rank, generate=True), rank) for rank in ranks[matrix_t])
    print('Computations for matrix type: ', matrix_t)
    for M, rank in matrices:
        print(padding(2) + 'Computations for rank', rank)

        exec_times[matrix_t][rank] = []
        exec_ranks[matrix_t][rank] = []
        rank_cp = rank

        while rank_cp >= 2:
```

```

rank_cp //= 2
print(padding(4) + 'Current rank:', rank_cp, end = ' ')

schur_complement(M, rank_cp, timer = main_timer)

exec_times[matrix_t][rank].append(main_timer.elapsed)
exec_ranks[matrix_t][rank].append(rank_cp)

print(f'{main_timer.elapsed:.5f}s')

# narysować wykresy

```

```

[23]: %matplotlib inline

for matrix_t in matrixtypes:
    for rank in ranks[matrix_t]:
        _, ax = plt.subplots(figsize=(12.7, 7))

        max_y = max(exec_times[matrix_t][rank])
        max_y += 0.1 * max_y
        plt.ylim(0, max_y)

        ax.scatter(
            exec_ranks[matrix_t][rank],
            exec_times[matrix_t][rank],
            label=f'{matrix_t}'
        )

        ax.plot(
            exec_ranks[matrix_t][rank],
            exec_times[matrix_t][rank],
            linestyle='--'
        )

        ax.set(
            xlabel='Stopień dopełnienia Schura',
            ylabel='Czas obliczeń [s]',
            title=f'Macierz stopnia {rank}, typu: {matrix_t}'
        )

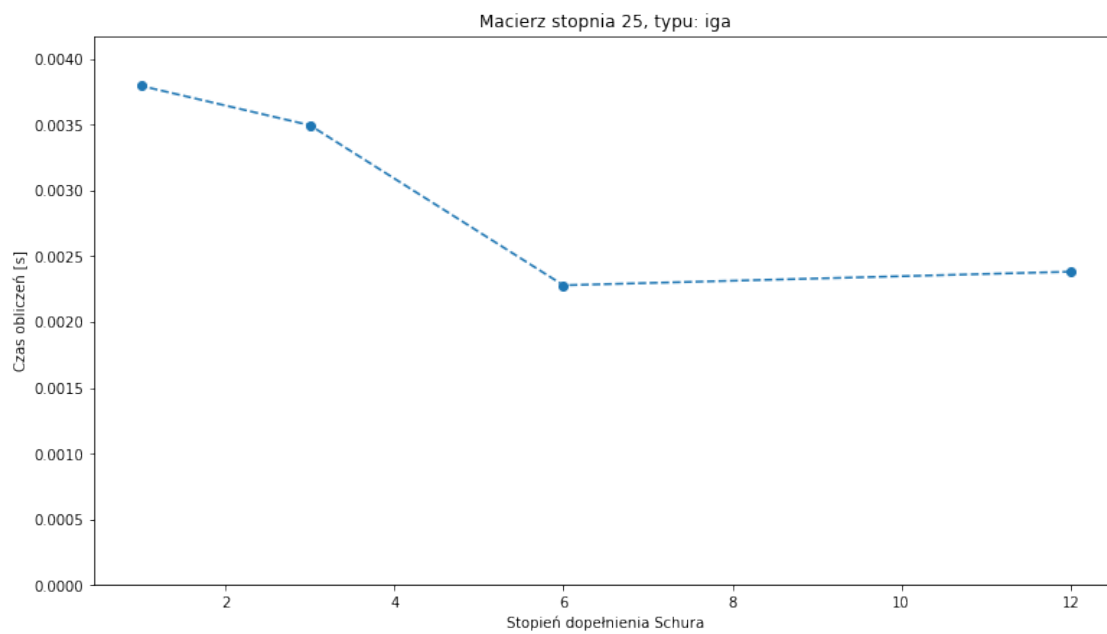
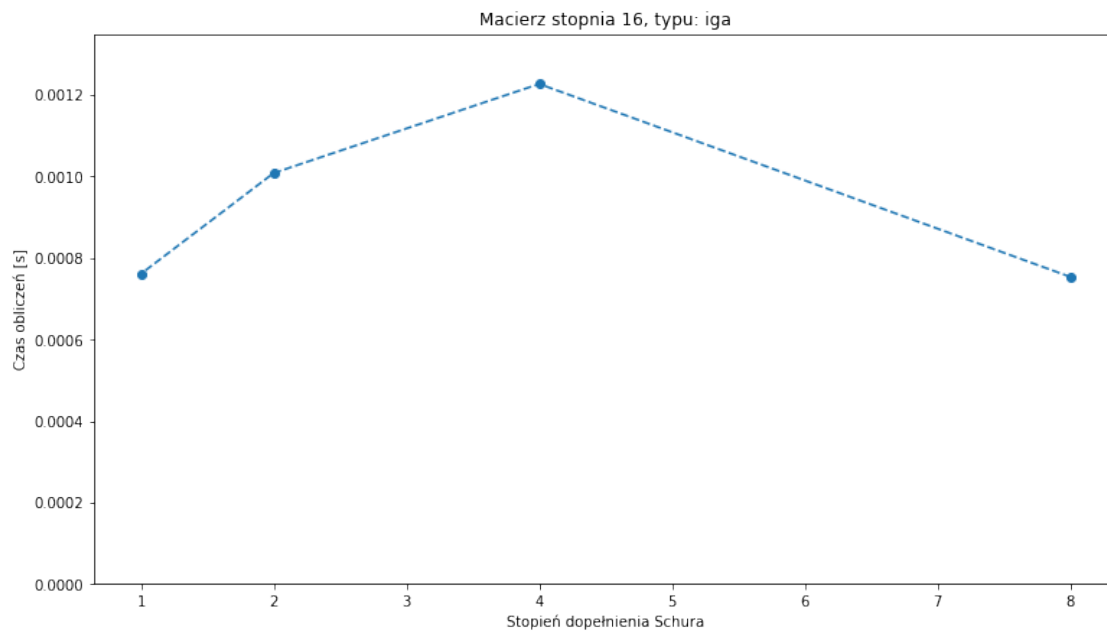
```

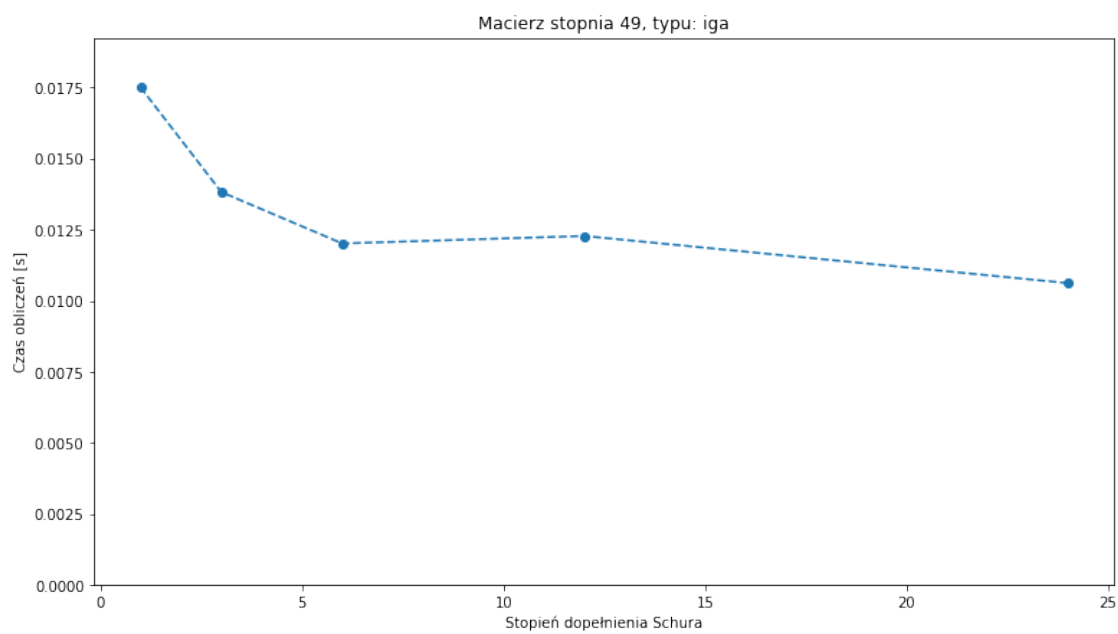
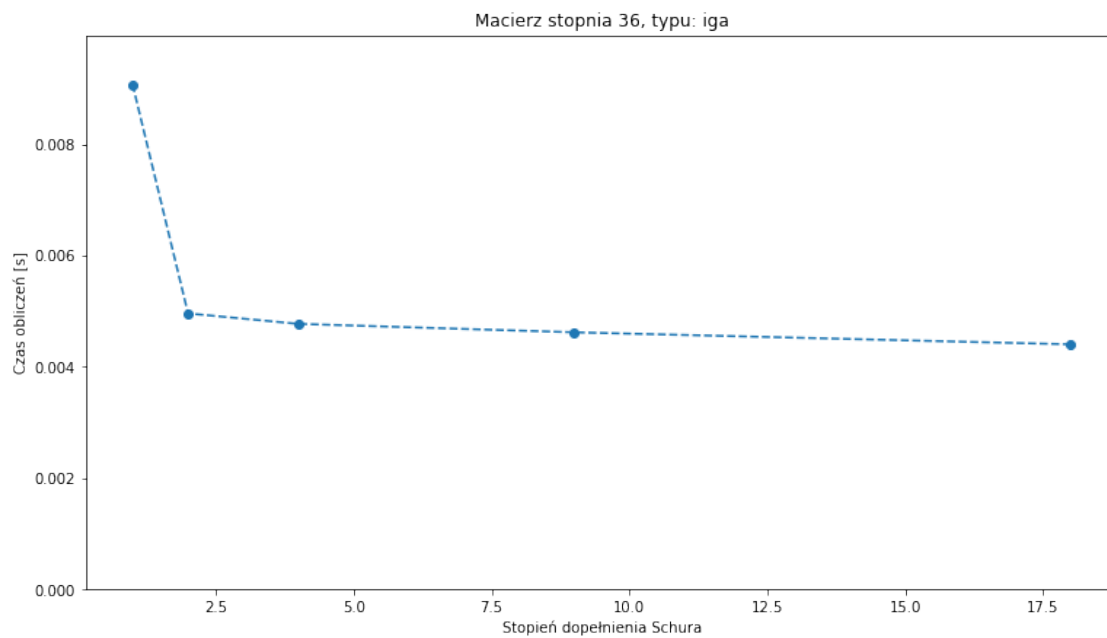
/tmp/ipykernel_186981/2887758238.py:5: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).

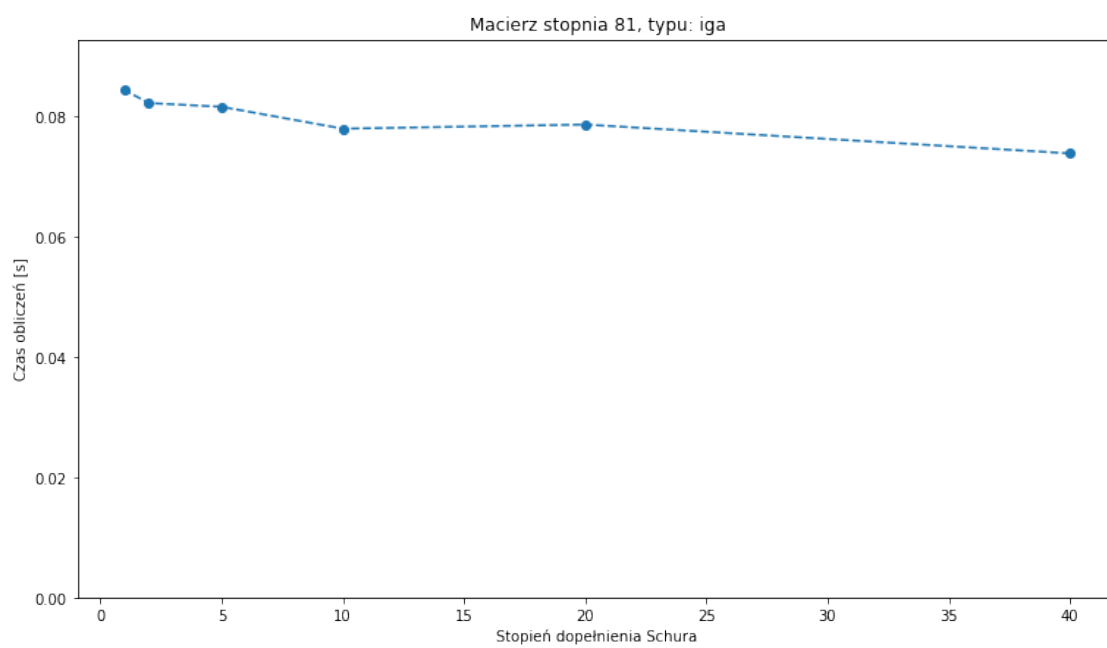
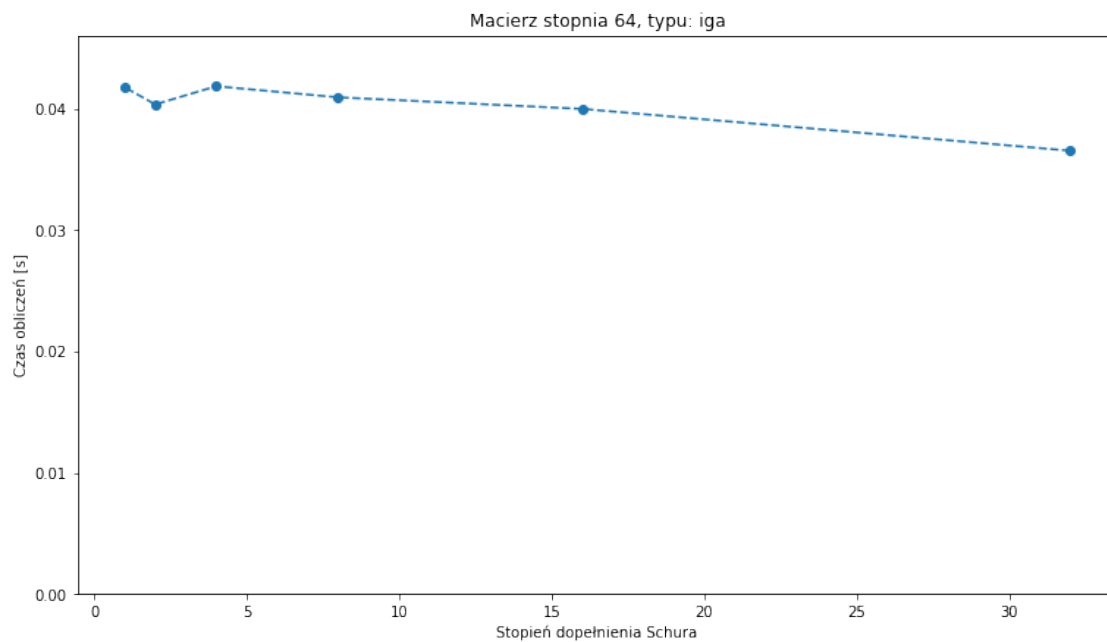
```

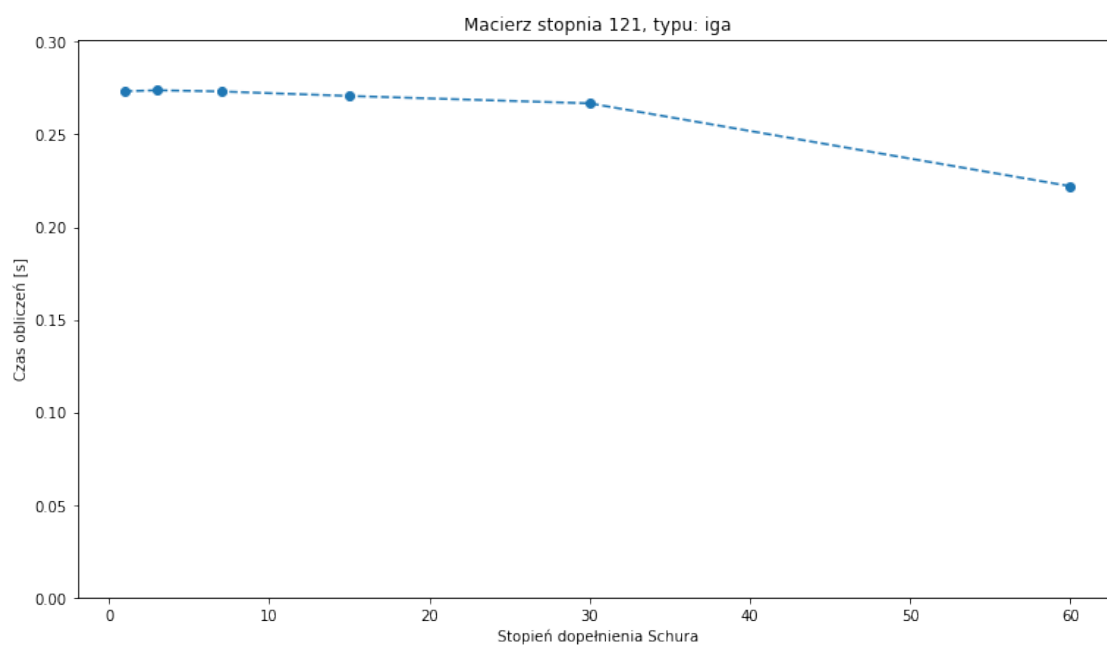
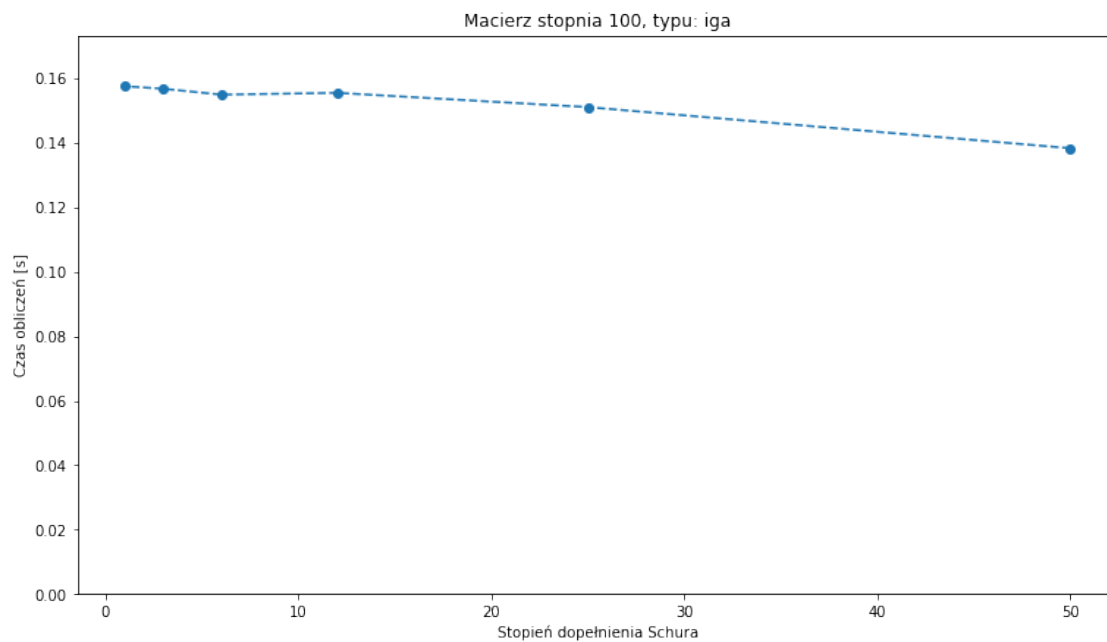
_, ax = plt.subplots(figsize=(12.7, 7))

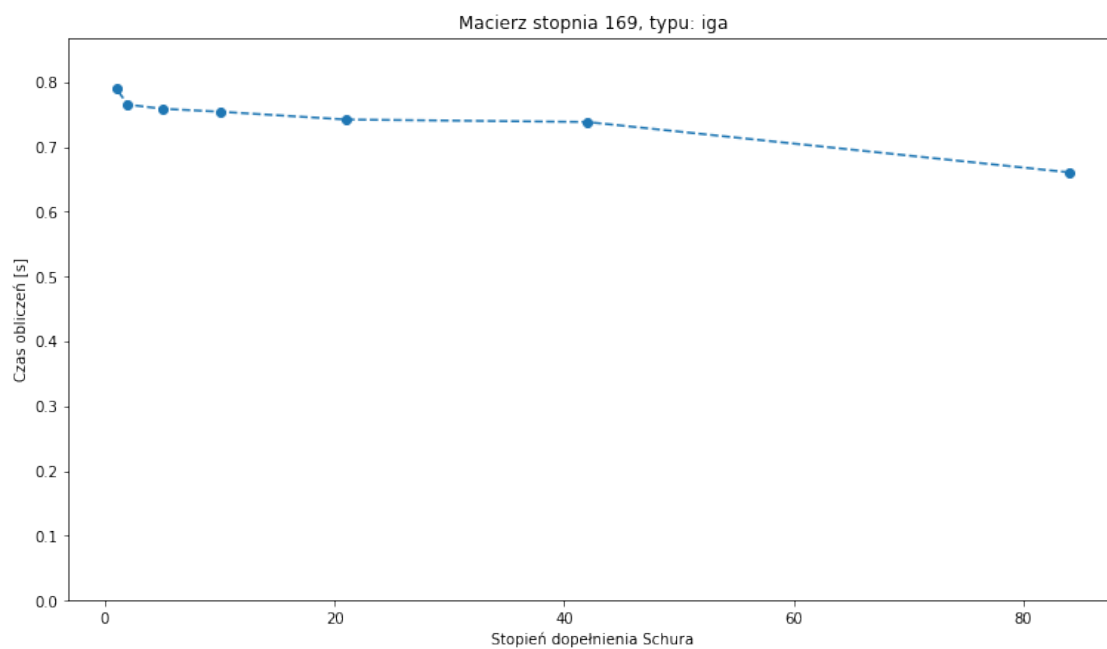
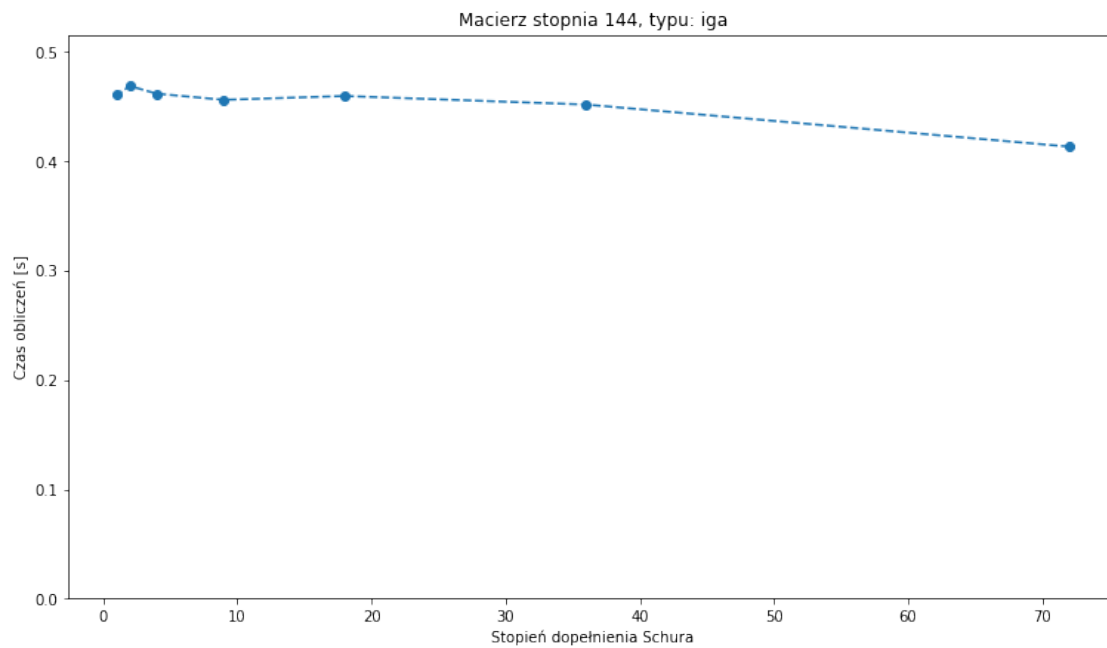
```

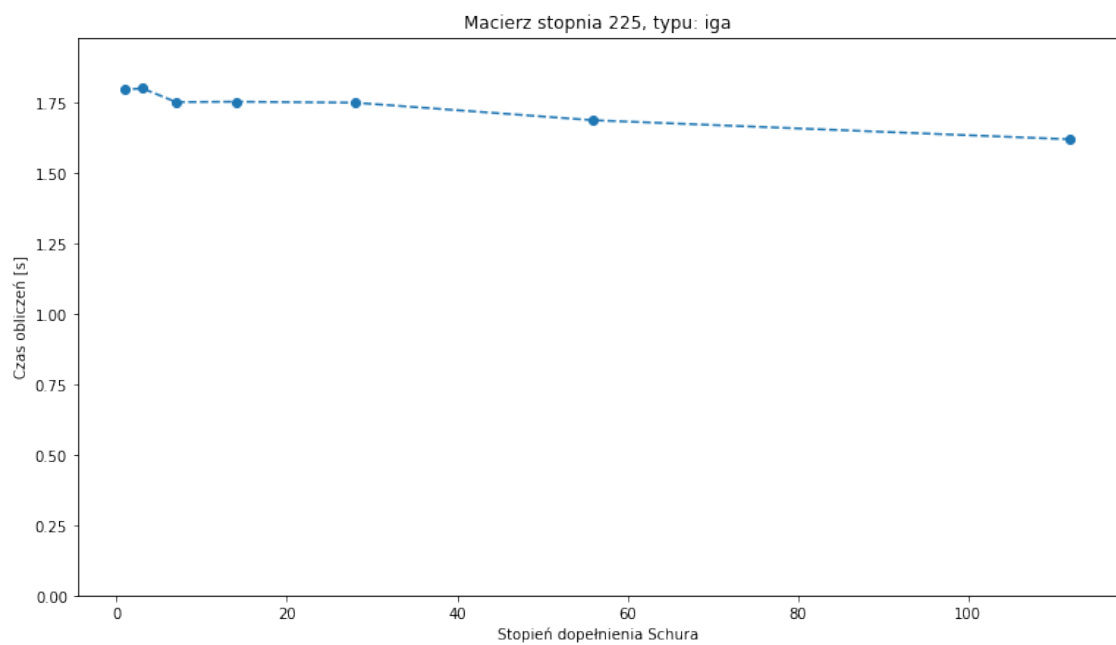
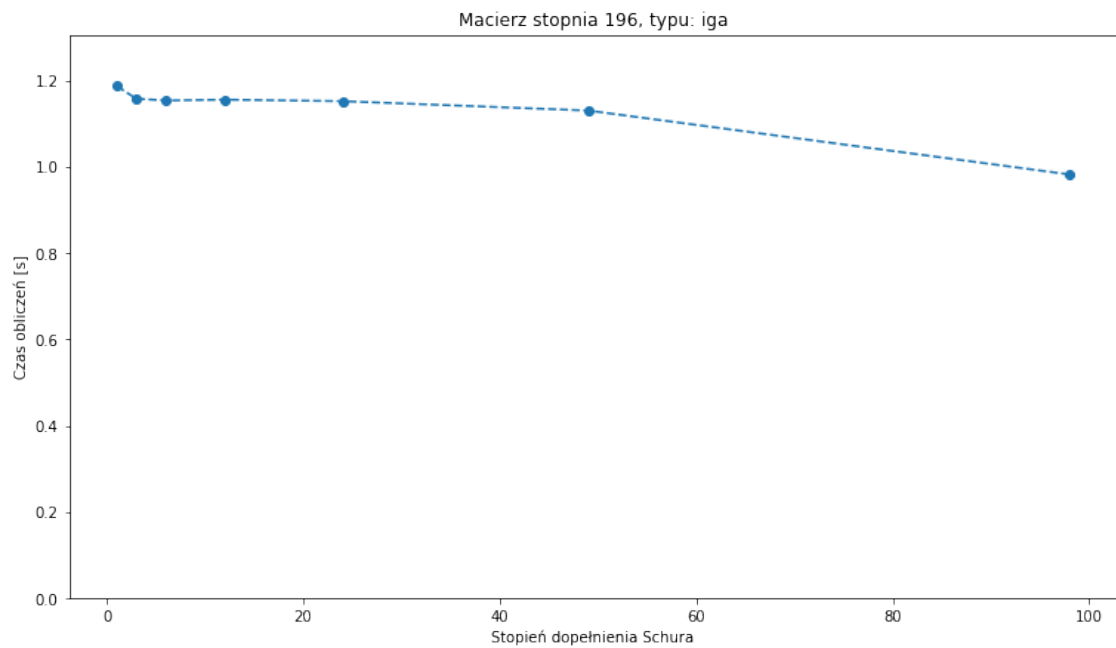


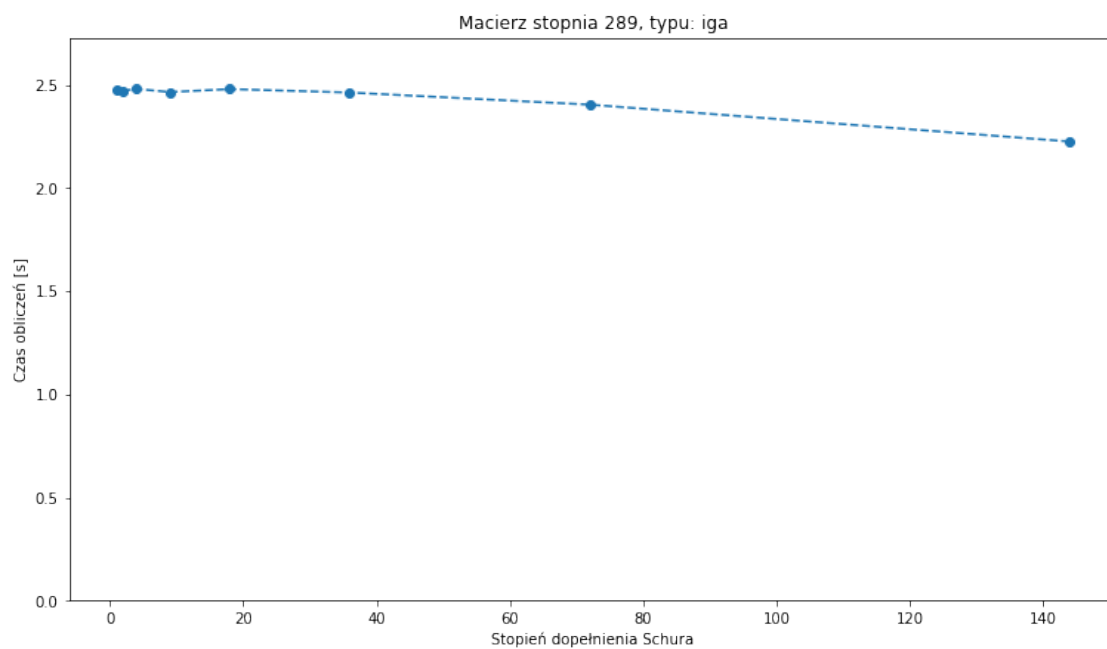
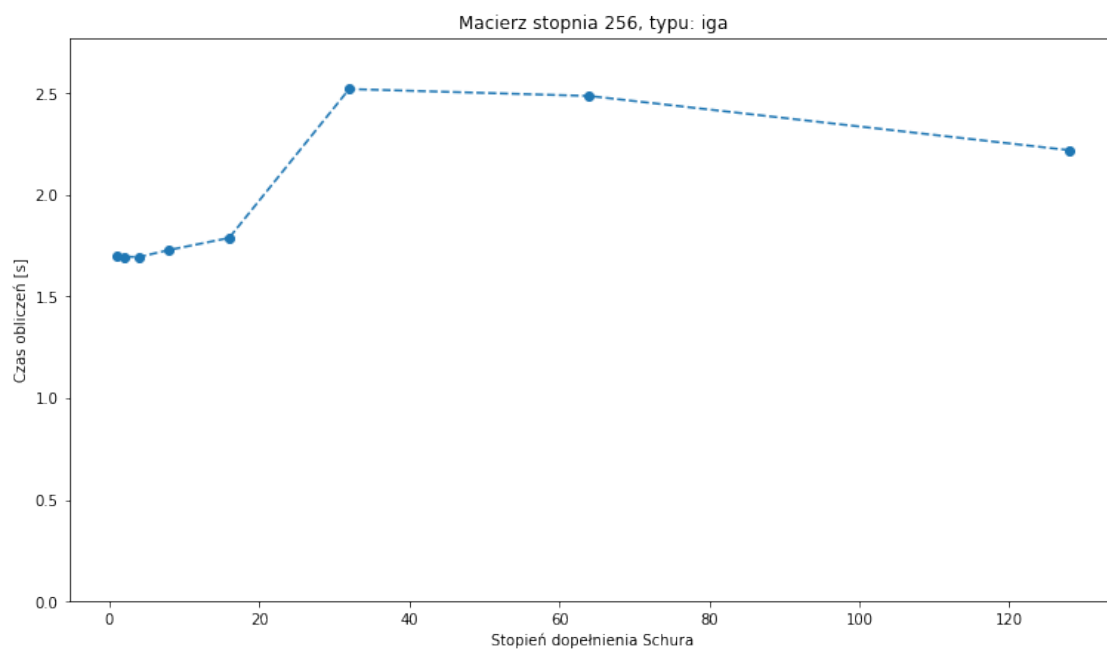


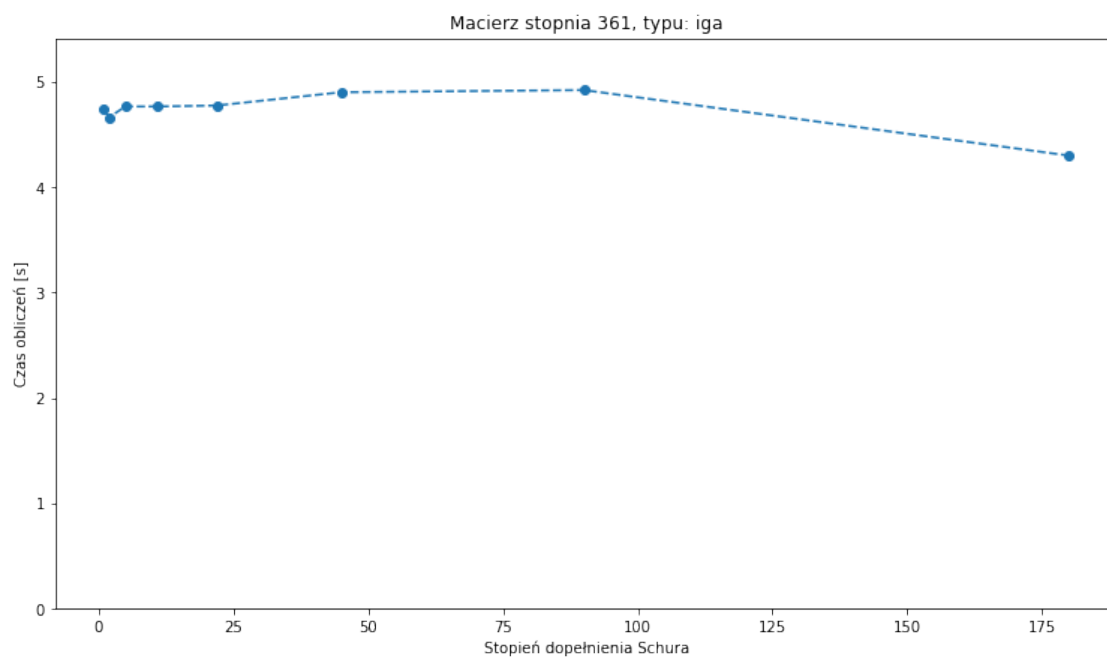
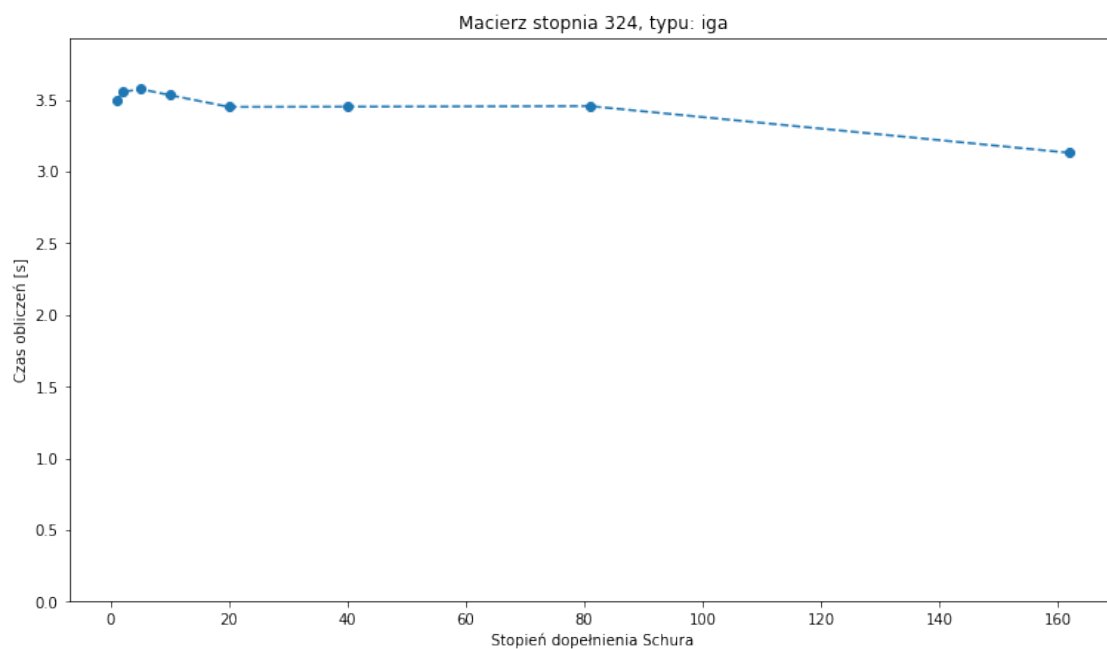


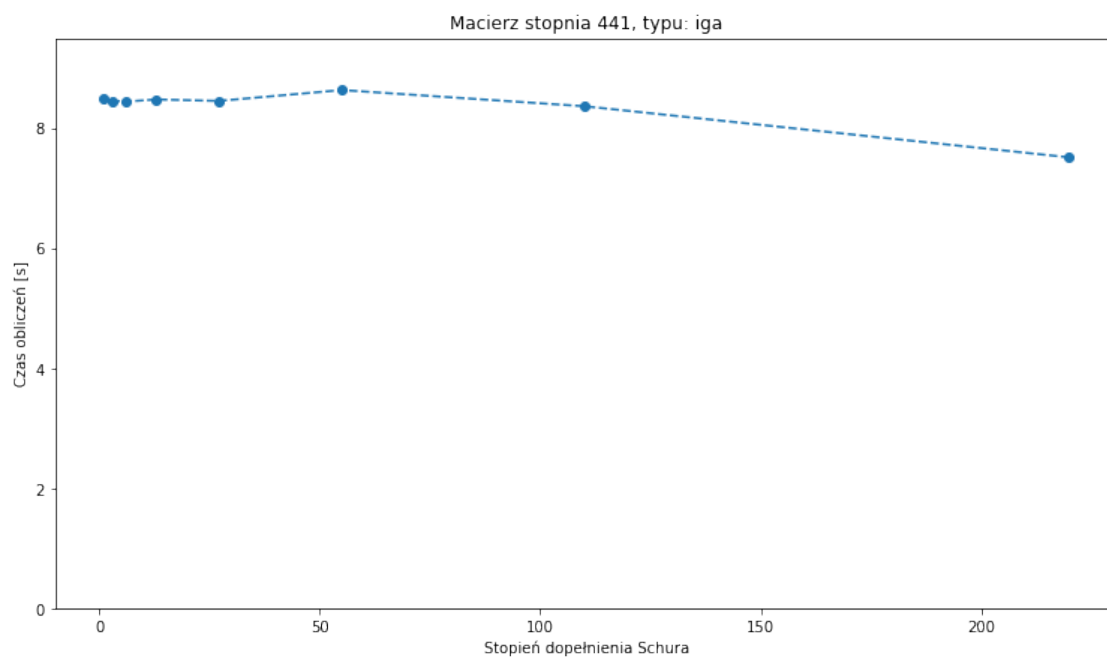
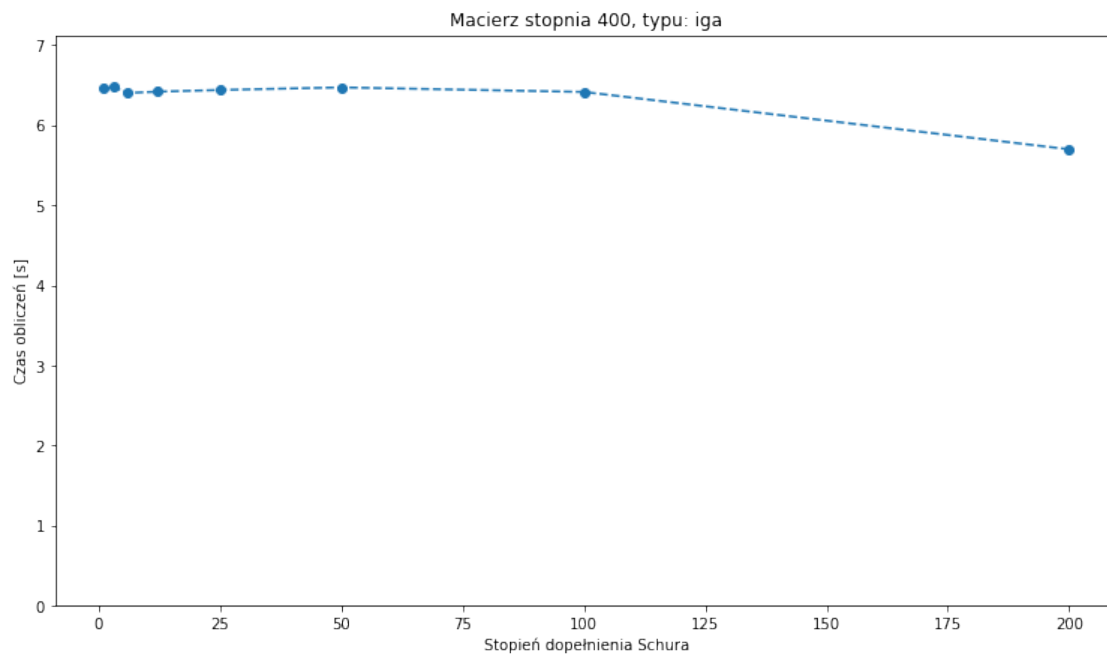


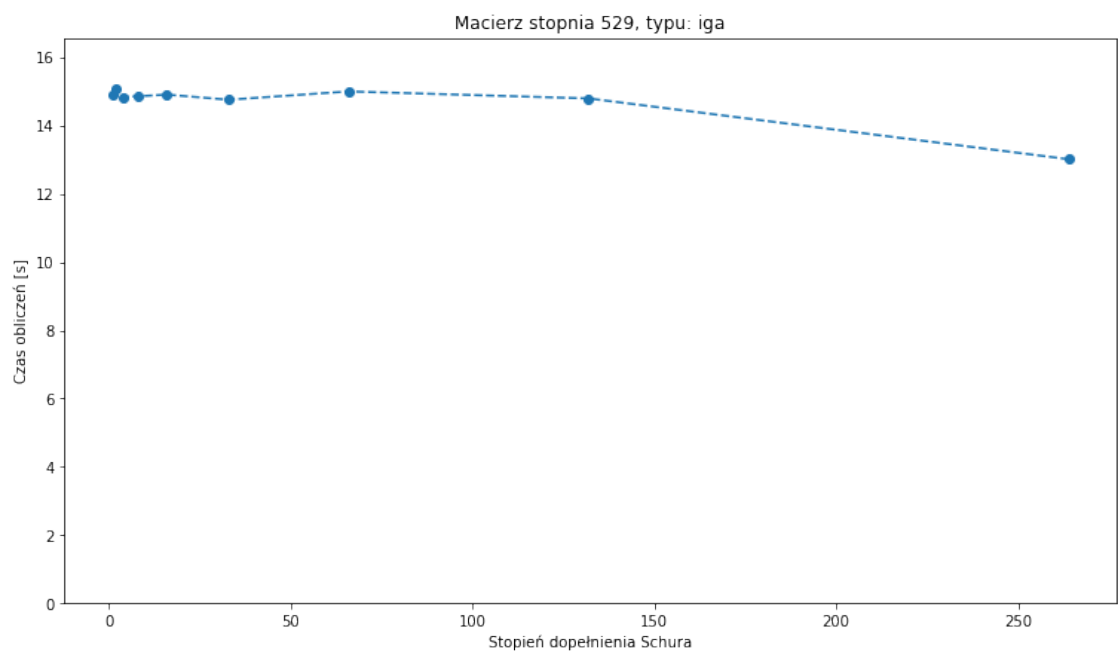
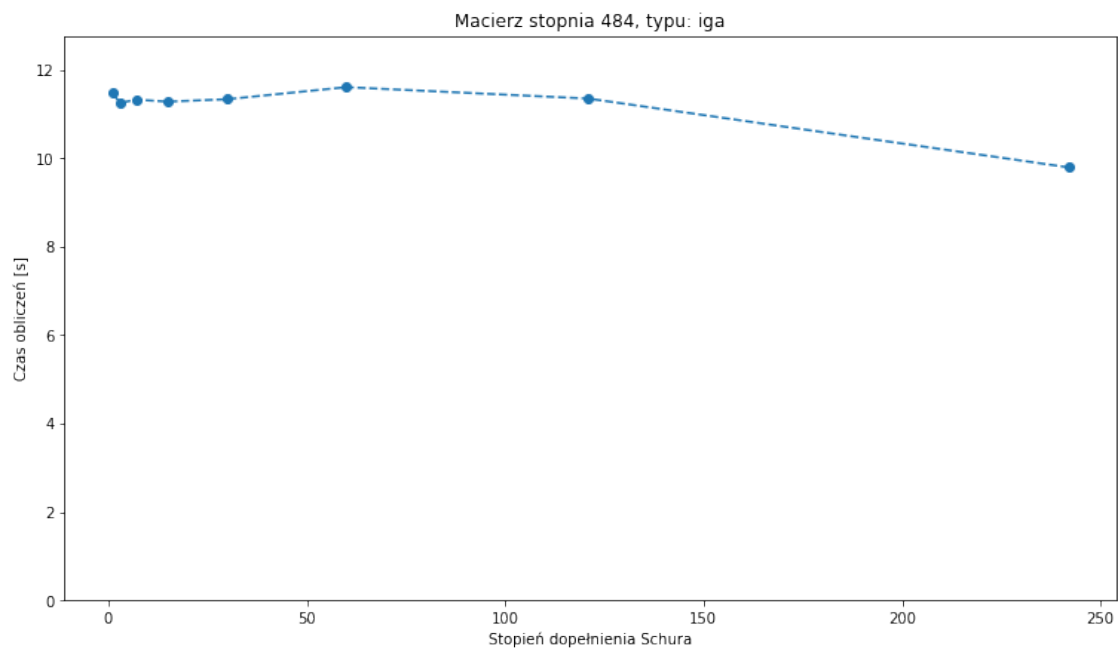


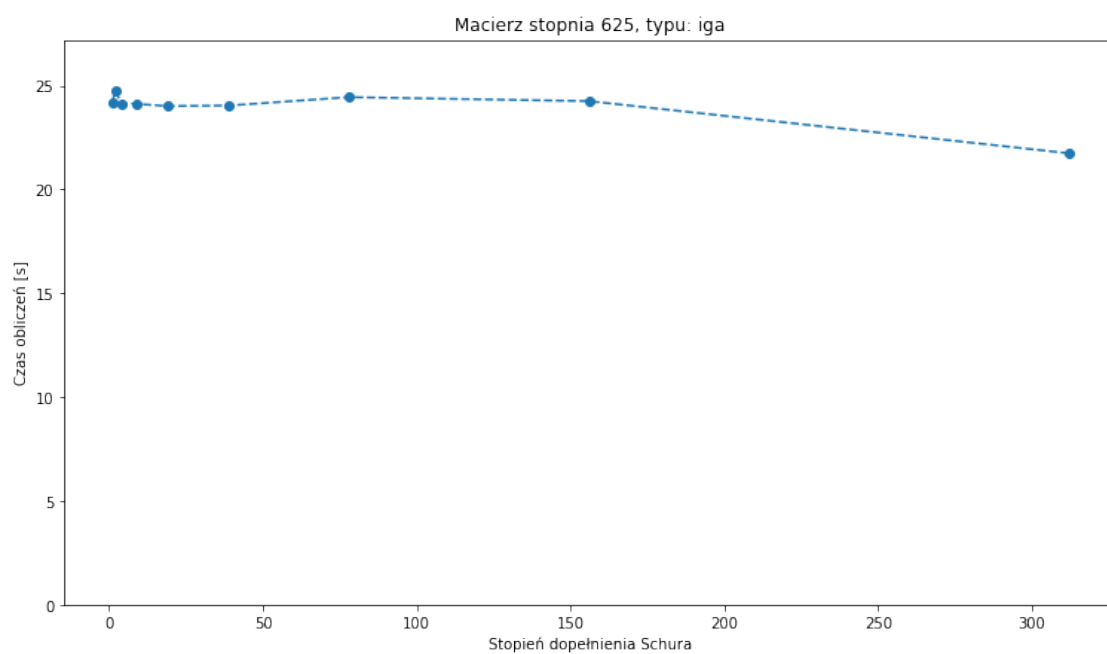
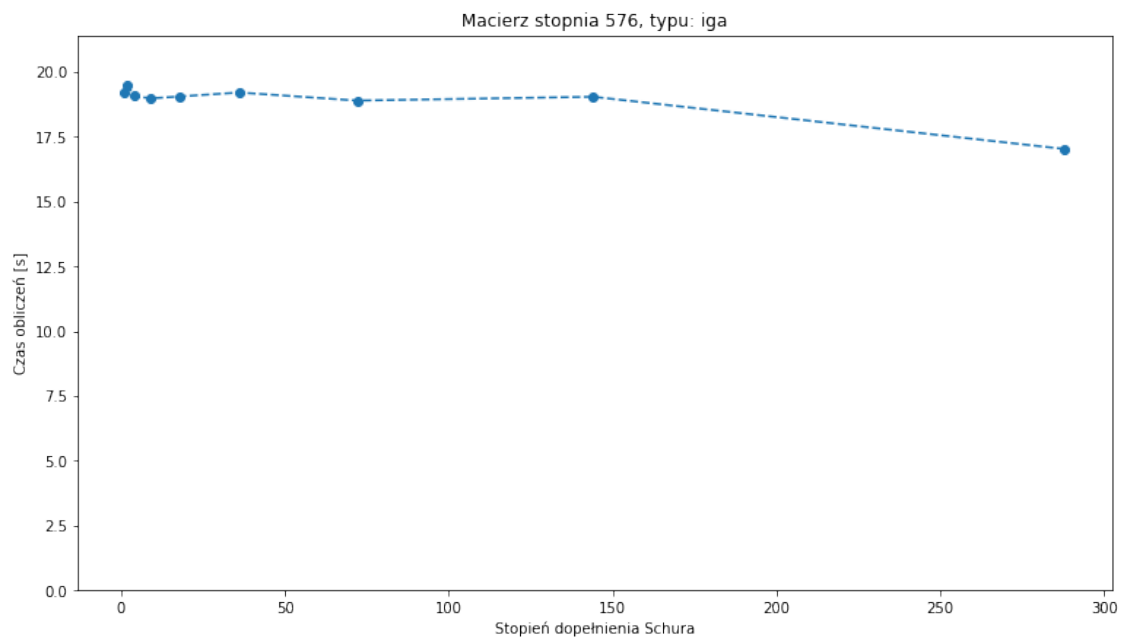


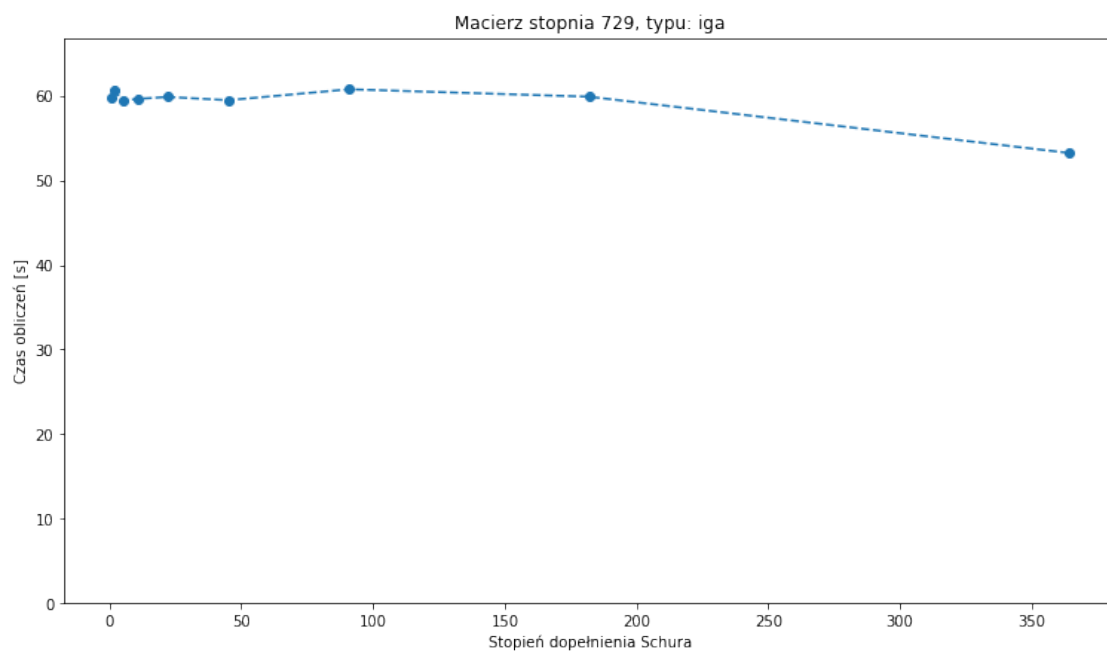
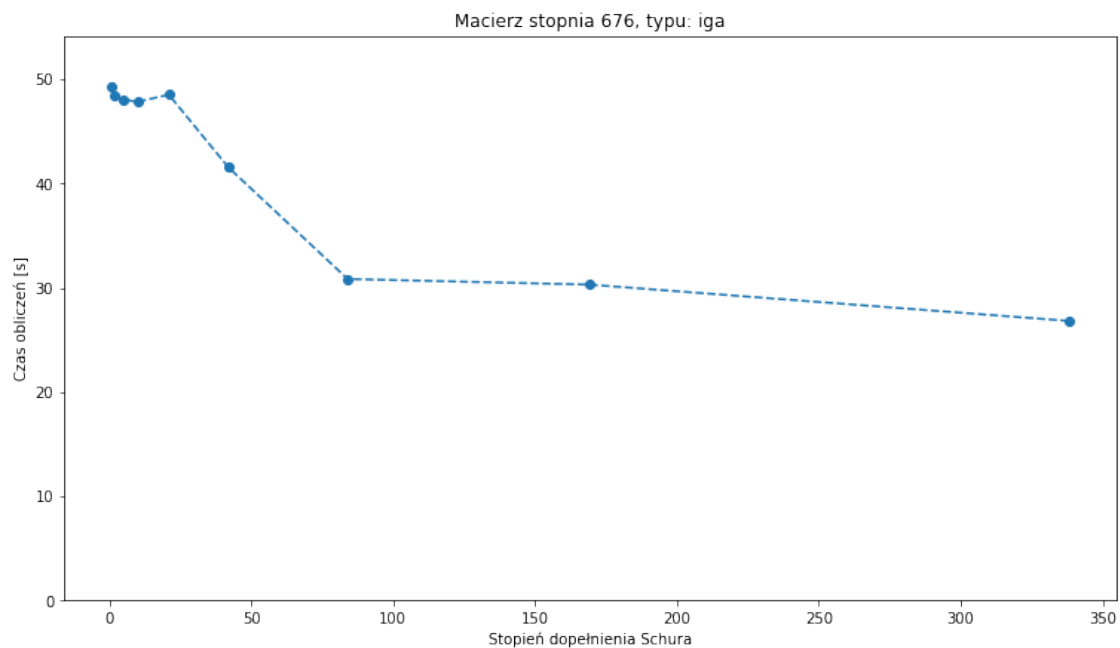


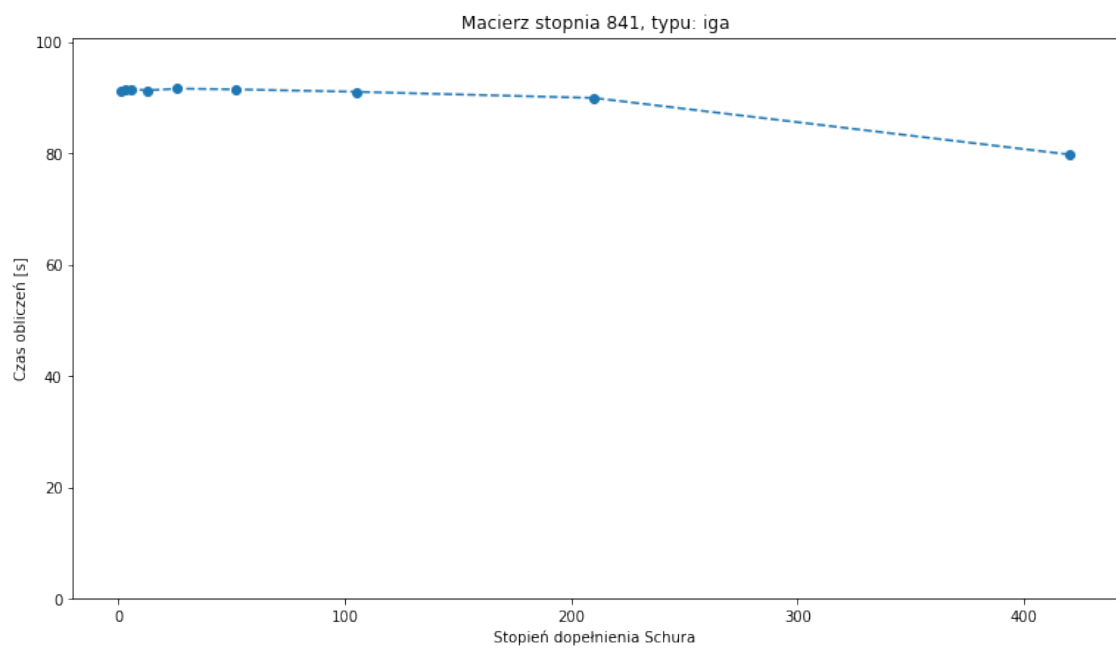
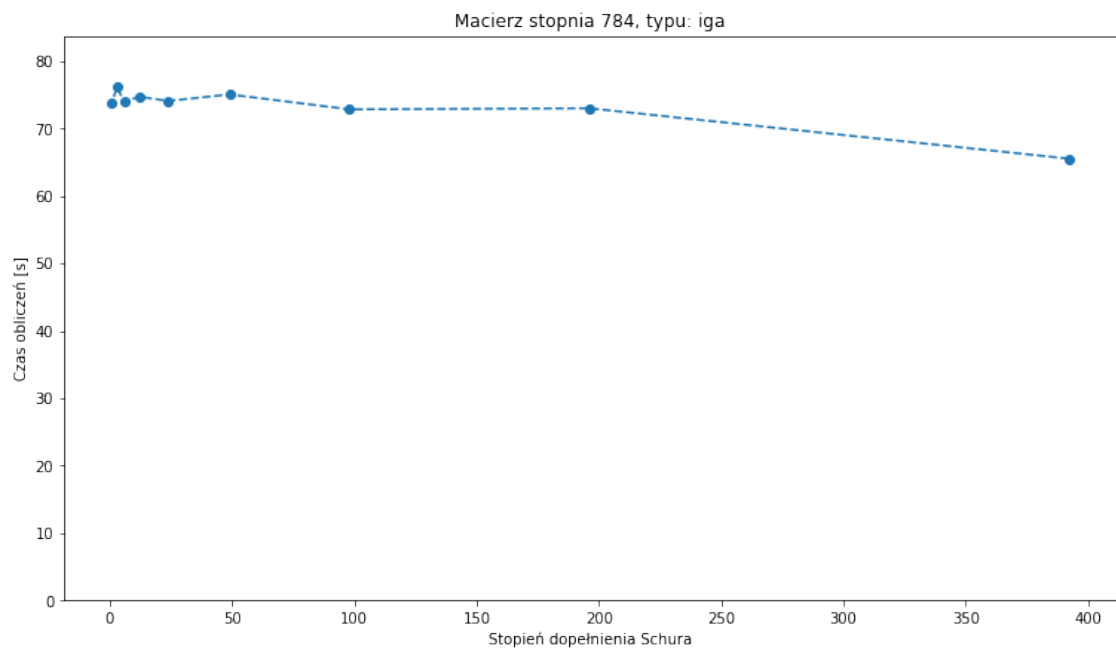


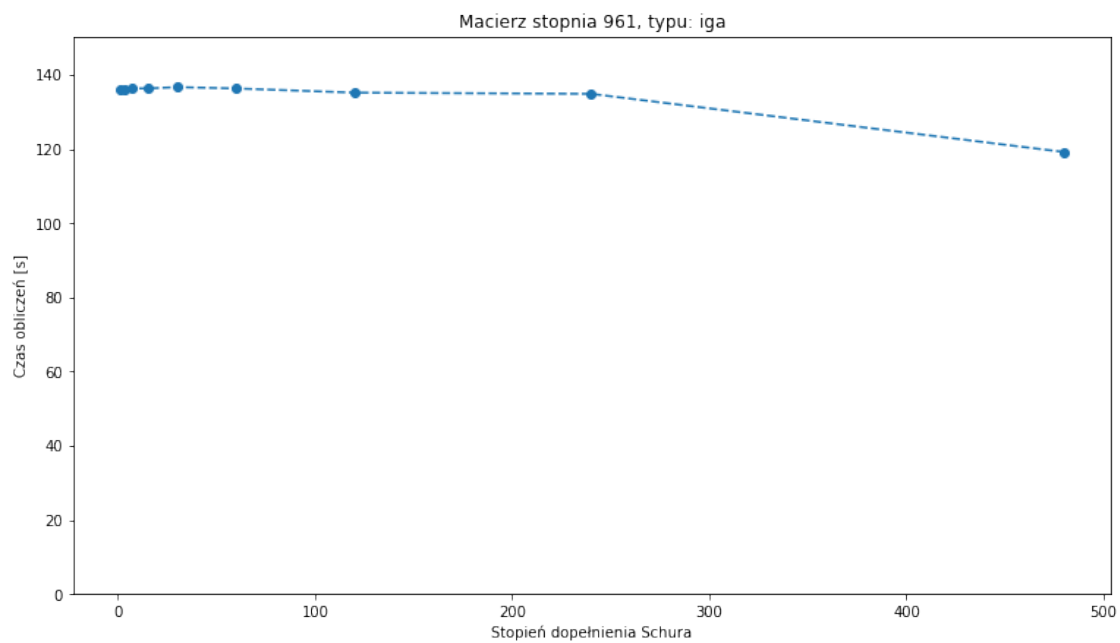
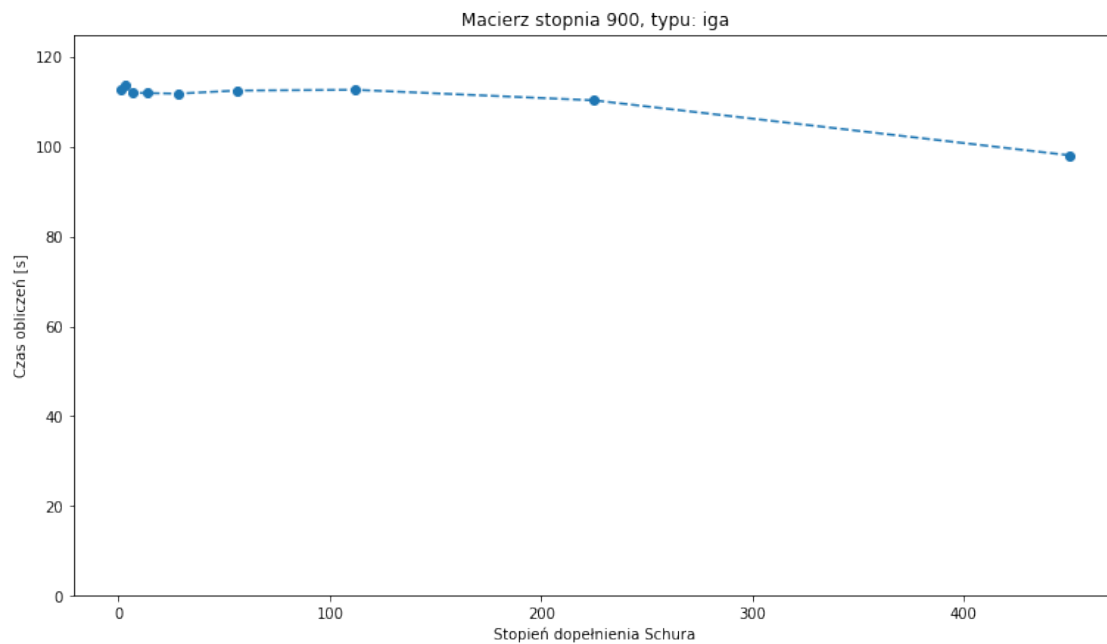


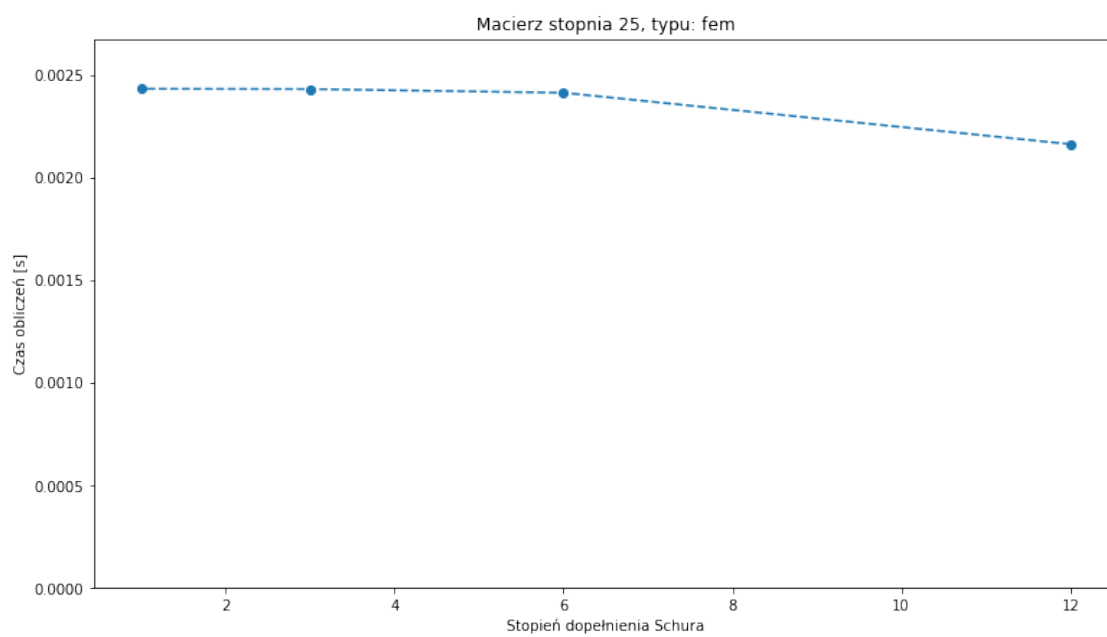
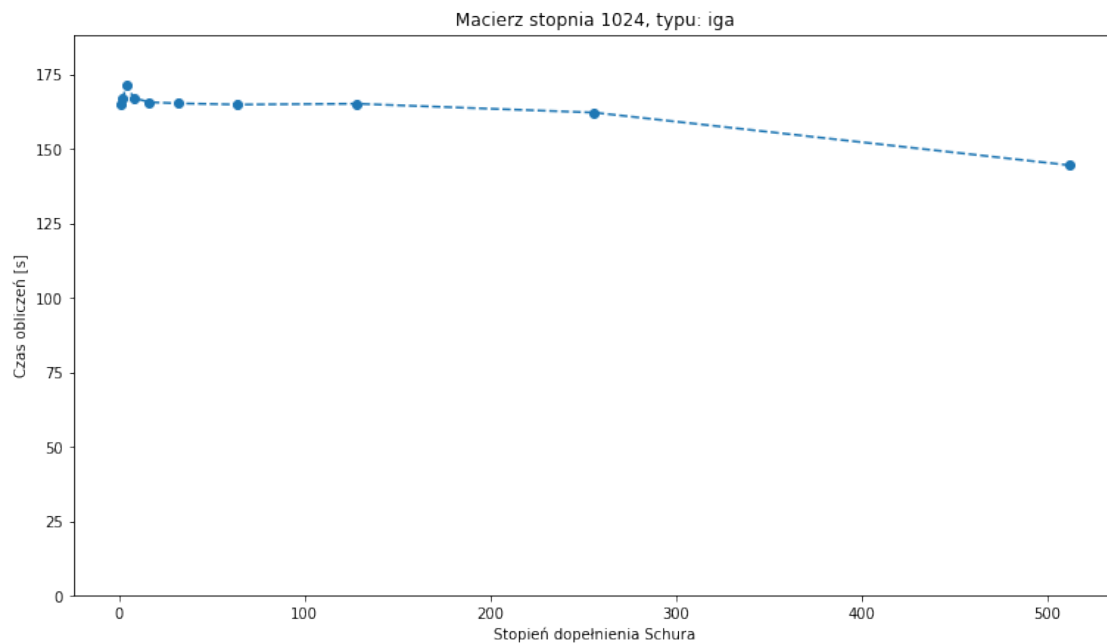


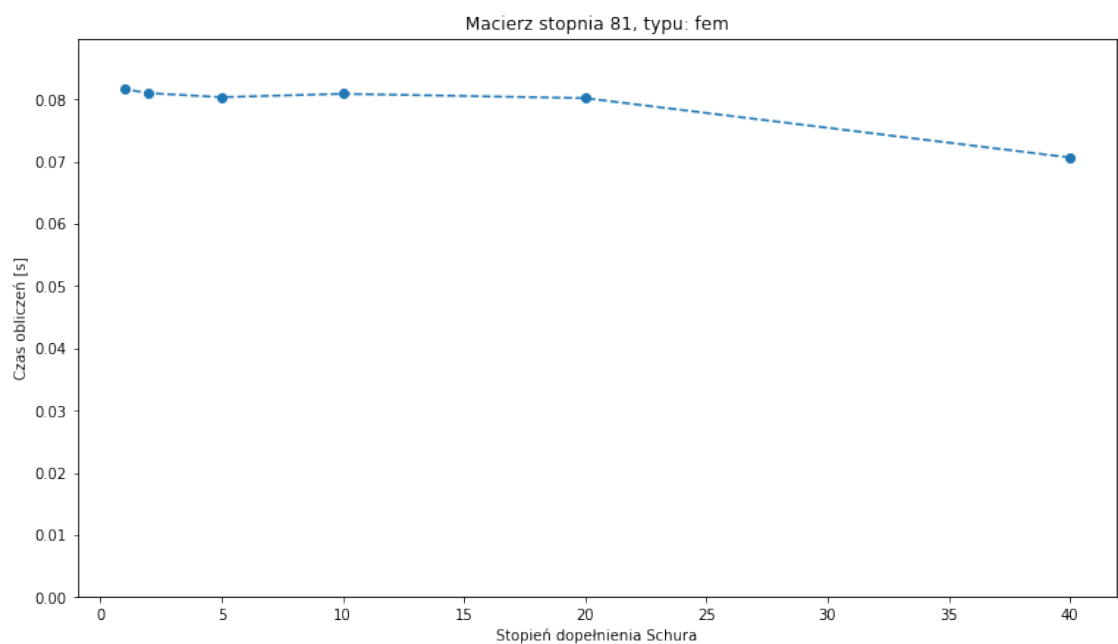
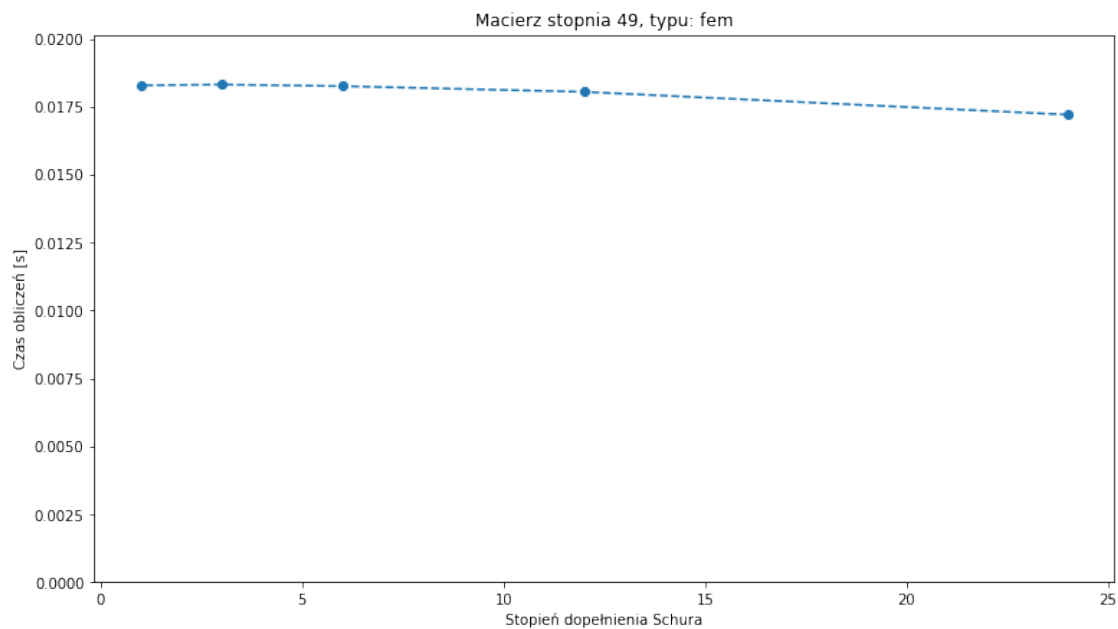


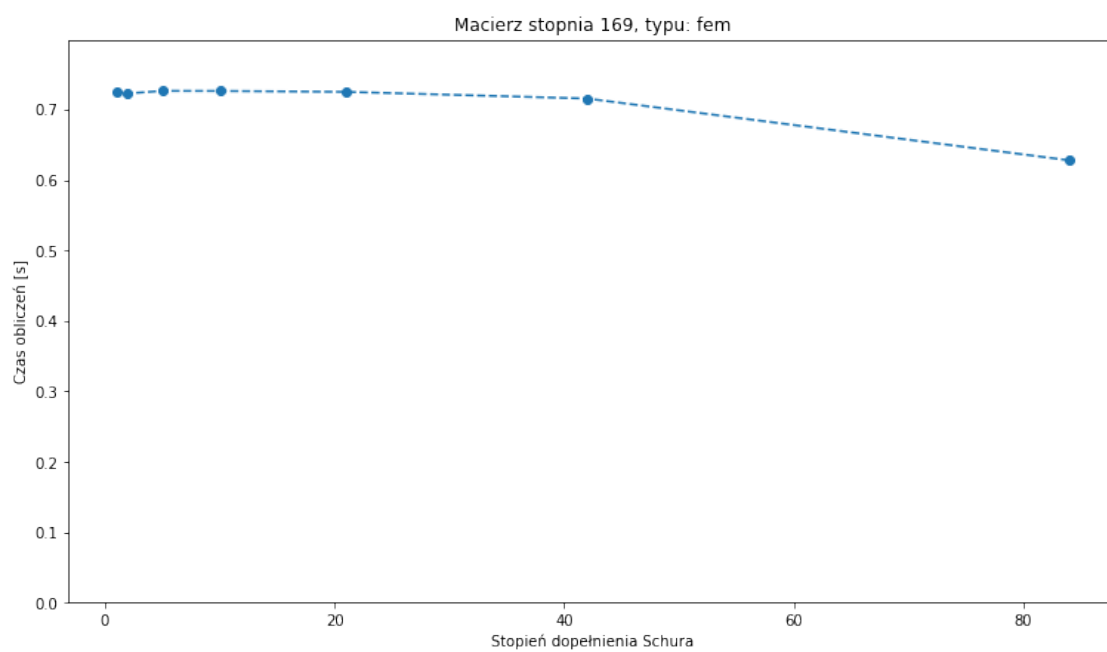
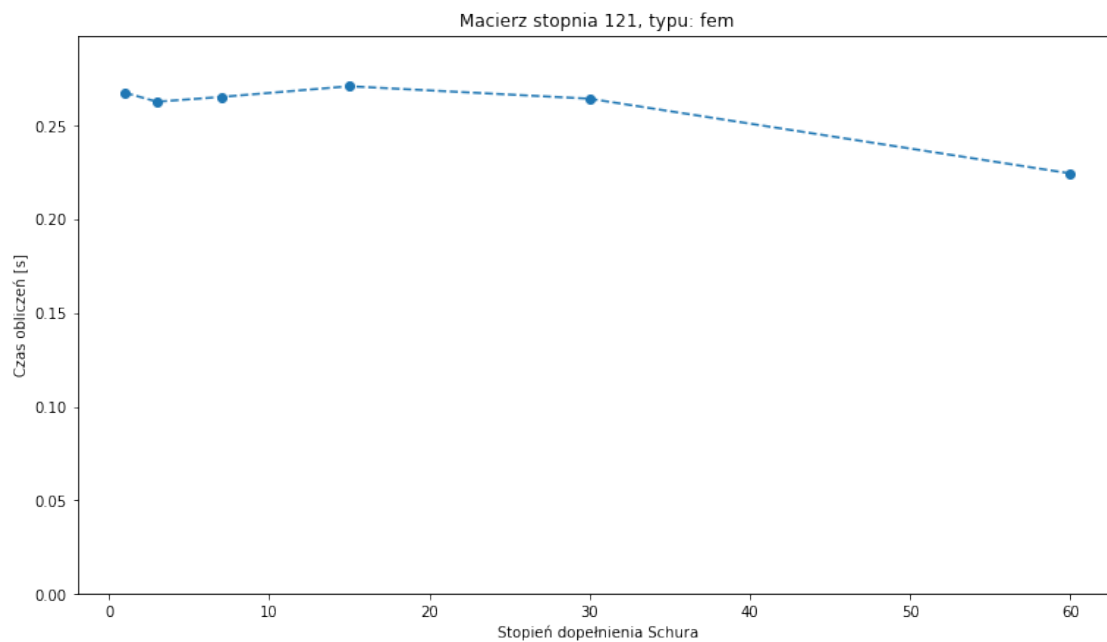


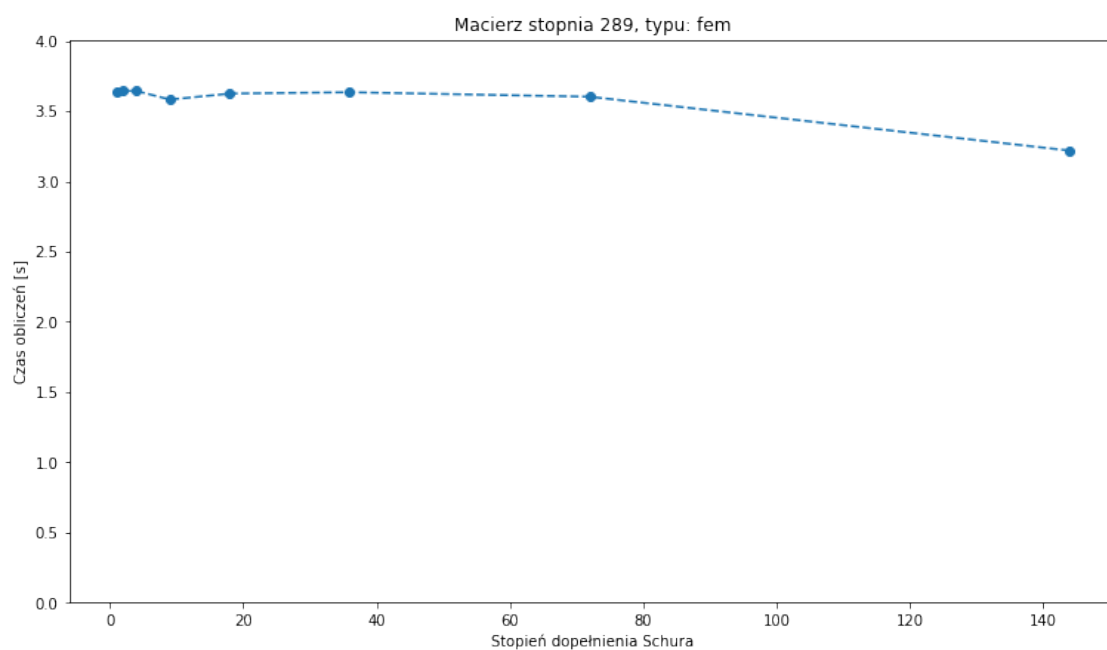
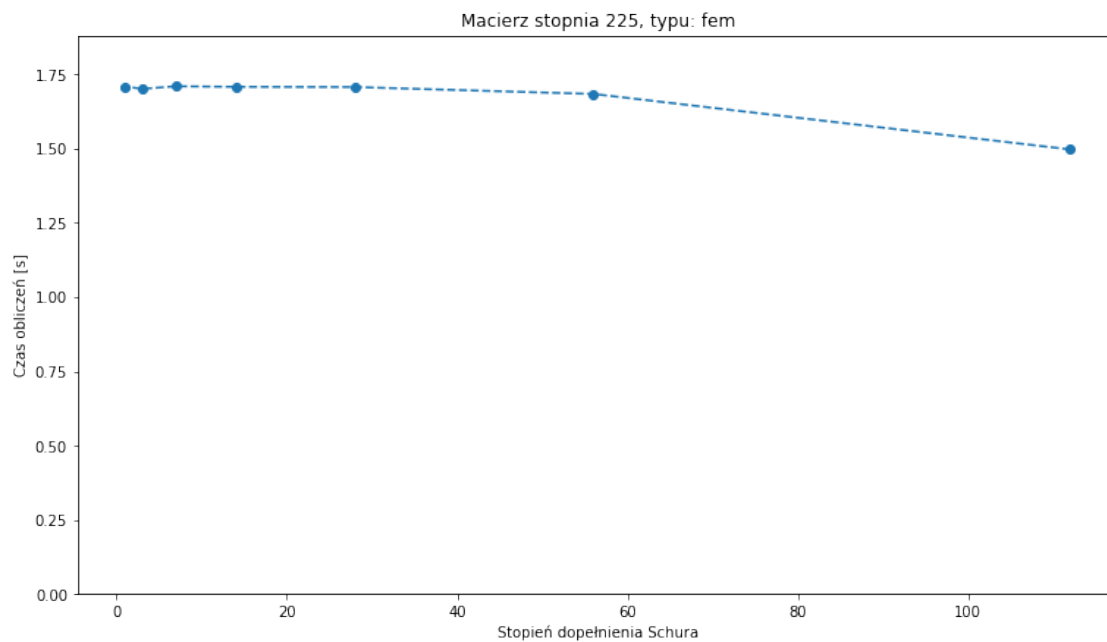


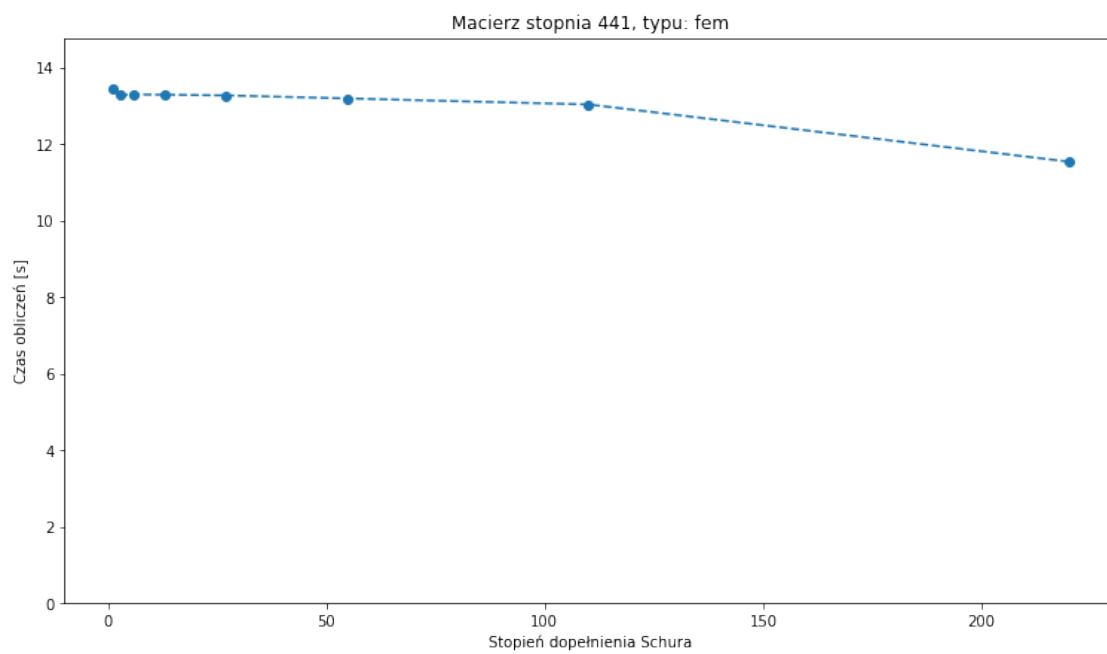
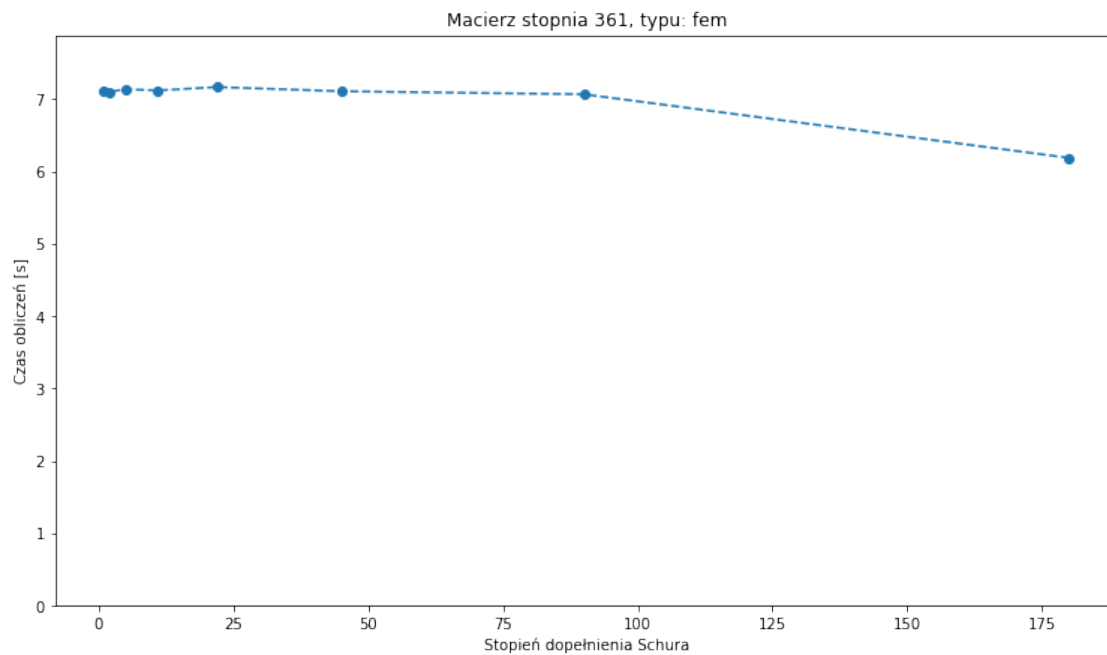


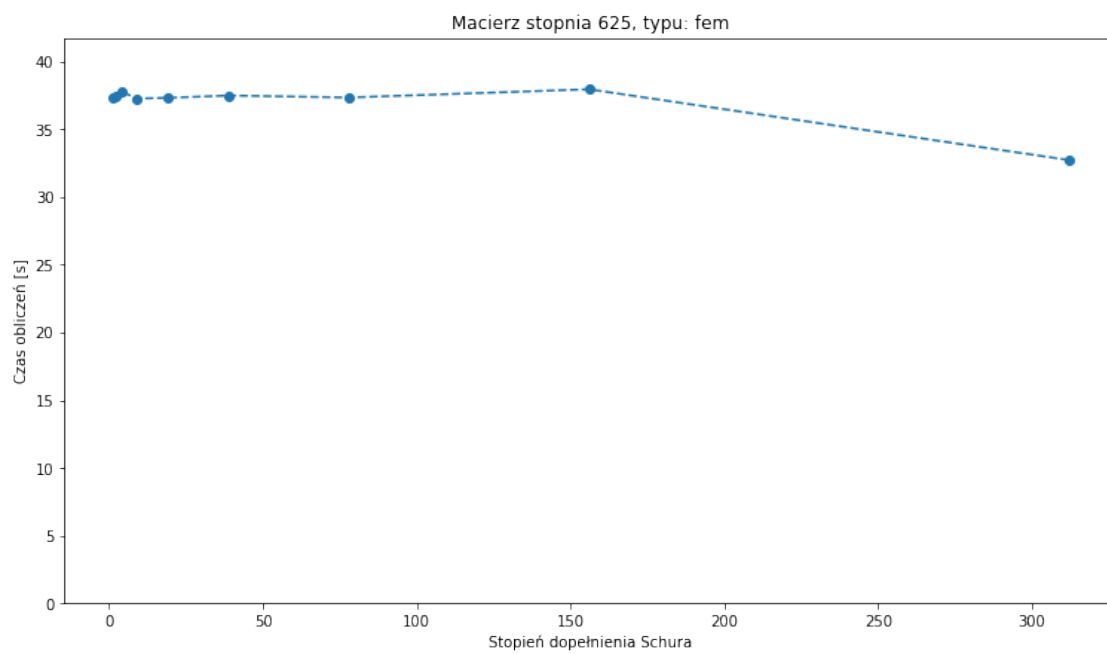
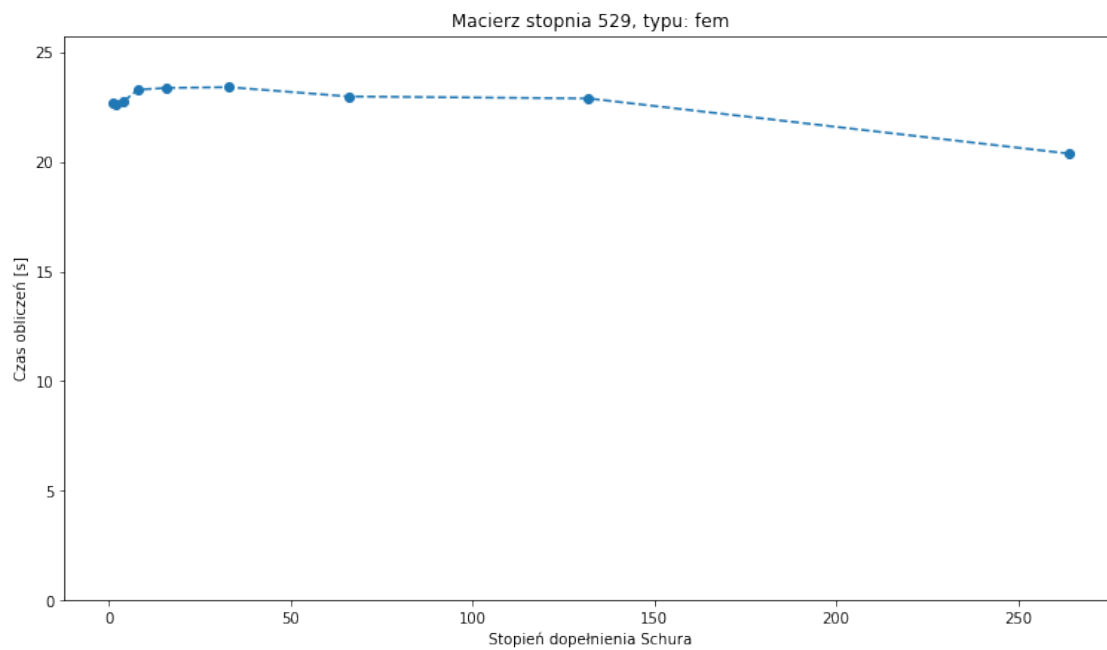


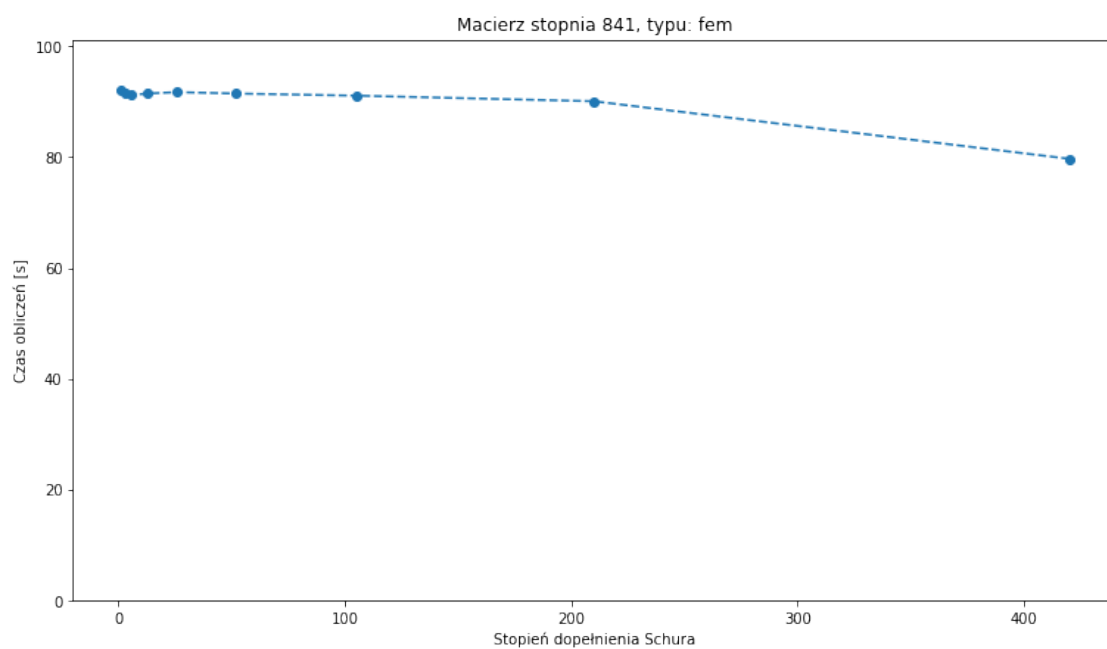
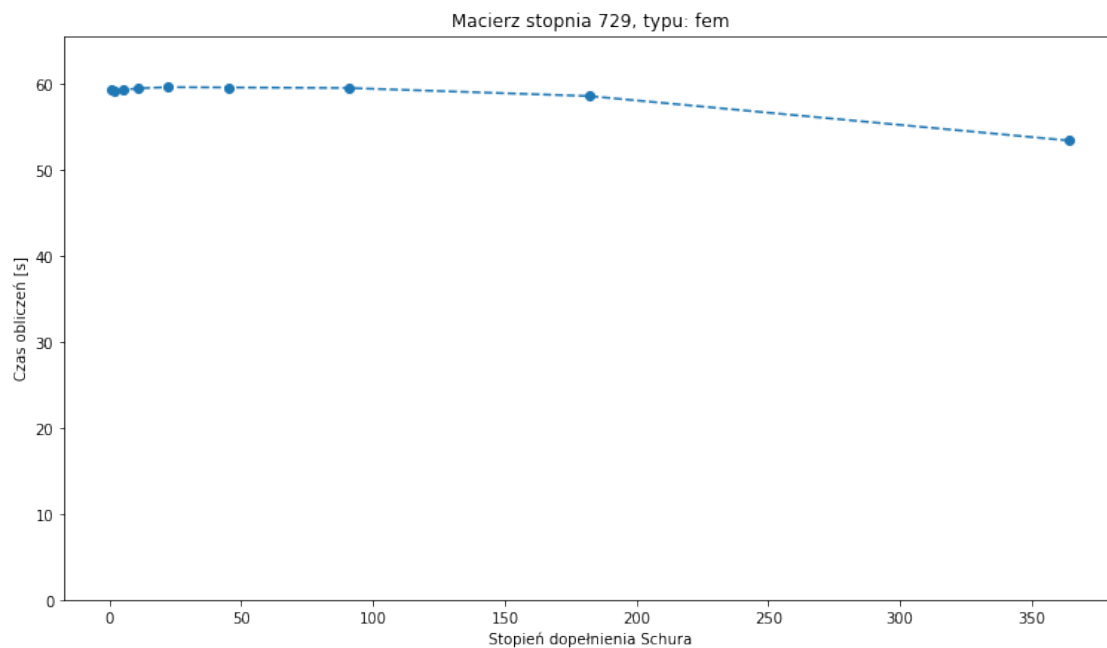


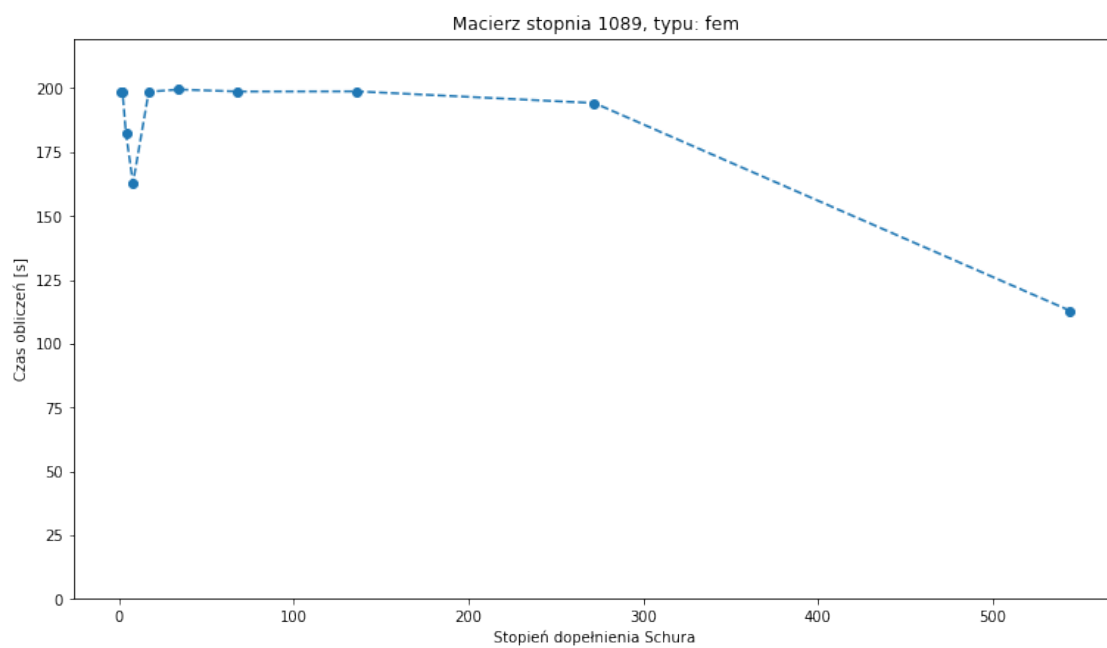
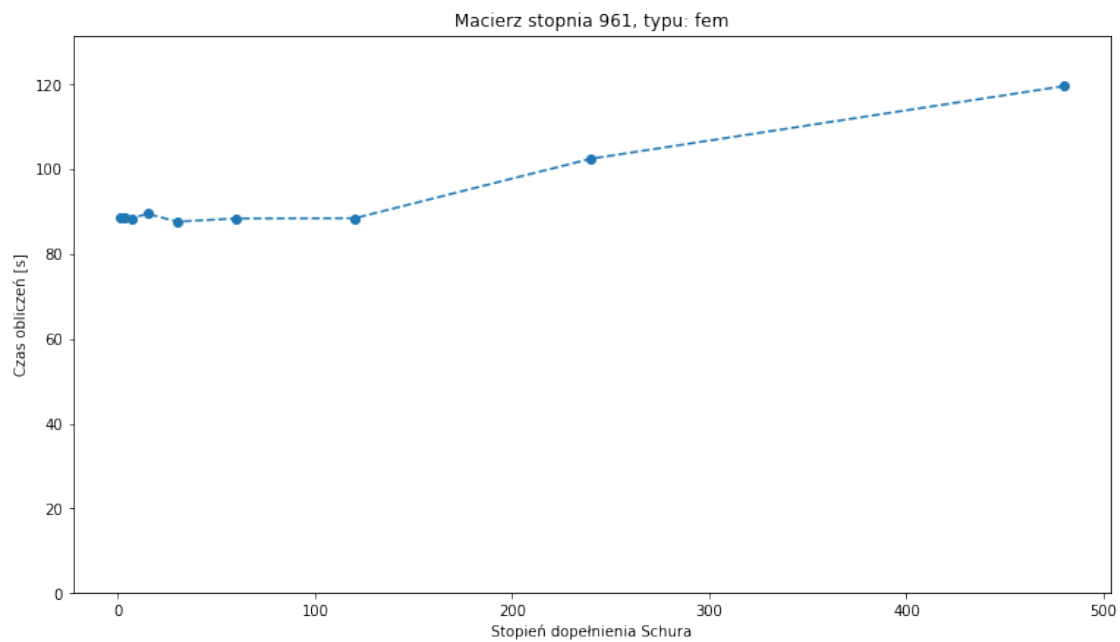












[]: