|  |  |  |  |
| --- | --- | --- | --- |
| MO\_03 | Romaniak Hubert | Informatyka niestacjonarna II rok | Semestr letni 2023/24 |

# Zadanie 1

## Metody różnic skończonych w języku Python

1. def progresive\_diff\_derivative(f, x, h):

2. return (f(x+h) - f(x)) / h

3.

4. def regresive\_diff\_derivative(f, x, h):

5. return (f(x) - f(x-h)) / h

6.

7. def central\_diff\_derivative(f, x, h):

8. return (f(x+h) - f(x-h)) / (2\*h)

9.

10. def central\_diff\_derivative\_2(f, x, h):

11. return (f(x-h) - 2\*f(x) + f(x+h)) / h\*\*2

## Dane

## Rozwiązanie za pomocą różnic skończonych i błąd w stosunku do wartości dokładnej

|  |  |  |  |
| --- | --- | --- | --- |
| pochodna i rodzaj różnicy | h | wartość | błąd |
| 1. pochodna za pomocą różnic wstecznych | 1.000000e-01 | 9.908364e-01 | 5.771545e-03 |
| 5.000000e-02 | 9.941358e-01 | 2.472168e-03 |
| 2.500000e-02 | 9.954755e-01 | 1.132454e-03 |
| 1.250000e-02 | 9.960677e-01 | 5.402952e-04 |
| 1. pochodna za pomocą różnic centralnych | 1.000000e-01 | 9.949478e-01 | 1.660183e-03 |
| 5.000000e-02 | 9.961927e-01 | 4.152014e-04 |
| 2.500000e-02 | 9.965041e-01 | 1.038101e-04 |
| 1.250000e-02 | 9.965820e-01 | 2.595313e-05 |
| 2. pochodna za pomocą różnic centralnych | 1.000000e-01 | 8.222725e-02 | 6.855698e-05 |
| 5.000000e-02 | 8.227866e-02 | 1.714353e-05 |
| 2.500000e-02 | 8.229152e-02 | 4.286150e-06 |
| 1.250000e-02 | 8.229473e-02 | 1.071554e-06 |

## Wykresy zależności od

Obraz zawierający tekst, linia, zrzut ekranu, Wykres

Opis wygenerowany automatycznie

Współczynnik nachylenia wykresu dla 1. pochodnej za pomocą różnic wstecznych wynosi około 1,14, a dla 1. i 2. pochodnej za pomocą różnic centralnych wynosi około 2,00.

Współczynniki te korelują z rzędami dokładności (wykładnikami przy wartości h) dla tych metod, gdzie błąd obcięcia dla 1. pochodnej za pomocą różnic wstecznych wynosi , a dla 1. i 2. pochodnej za pomocą różnic centralnych - .

# Zadanie 2

## Metoda pośrednia Eulera w języku Python, używając biblioteki *numpy*

1. from numpy import sin, cos, exp, square, pi, arange, abs

2.

3. def y\_next\_getter\_getter(a, dt):

4. def y\_next\_getter(y\_previous, t):

5. return (y\_previous + dt \* sin(pi \* t)) / (1 + a \* dt)

6. return y\_next\_getter

7.

8. def exact\_solution\_getter(a):

9. def exact\_solution(t):

10. numerator = pi \* exp(-a\*t) - pi \* cos(pi\*t) + a \* sin(pi\*t)

11. denominator = square(pi) + square(a)

12. return numerator / denominator

13. return exact\_solution

14.

15. def implicit\_euler\_getter(a, t\_min, t\_max, y0):

16. def implicit\_euler(N):

17. dt = (t\_max - t\_min) / N

18. exact\_solution = exact\_solution\_getter(a)

18. y\_next\_getter = y\_next\_getter\_getter(a, dt)

19. current\_t\_getter = lambda i: t\_min + dt \* i

20. ts = [current\_t\_getter(0)]

21. ys = [y0]

22. errors = [abs(exact\_solution(ts[0]) - ys[0])]

23. for i in arange(1, N + 1):

24. ts.append(current\_t\_getter(i))

25. ys.append(y\_next\_getter(ys[i - 1], ts[i]))

26. errors.append(abs(exact\_solution(ts[i]) - ys[i]))

27. return array(ts), array(ys), array(errors), dt

28. return implicit\_euler

## Problem początkowy

## Dane i ilość iteracji N

## Ścisłe rozwiązanie

### Sprawdzenie rozwiązania dla

### Sprawdzenie rozwiązania w ogólnym przypadku

## Kolejne kroki iteracyjne w metodzie pośredniej Eulera

|  |  |  |  |
| --- | --- | --- | --- |
| n | t | y | error |
| 0 | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 |
| 1 | 7.142857e-02 | 1.478544e-02 | 7.000909e-03 |
| 2 | 1.428571e-01 | 4.258339e-02 | 1.258963e-02 |
| 3 | 2.142857e-01 | 8.104035e-02 | 1.659683e-02 |
| 4 | 2.857143e-01 | 1.273353e-01 | 1.892209e-02 |
| 5 | 3.571429e-01 | 1.783165e-01 | 1.953981e-02 |
| 6 | 4.285714e-01 | 2.306551e-01 | 1.850141e-02 |
| 7 | 5.000000e-01 | 2.810080e-01 | 1.593345e-02 |
| 8 | 5.714286e-01 | 3.261821e-01 | 1.203204e-02 |
| 9 | 6.428571e-01 | 3.632902e-01 | 7.053620e-03 |
| 10 | 7.142857e-01 | 3.898933e-01 | 1.302678e-03 |
| 11 | 7.857143e-01 | 4.041194e-01 | 4.882946e-03 |
| 12 | 8.571429e-01 | 4.047545e-01 | 1.114854e-02 |
| 13 | 9.285714e-01 | 3.913012e-01 | 1.713983e-02 |
| 14 | 1.000000e+00 | 3.640012e-01 | 2.252037e-02 |
| 15 | 1.071429e+00 | 3.238203e-01 | 2.698801e-02 |
| 16 | 1.142857e+00 | 2.723987e-01 | 3.028972e-02 |
| 17 | 1.214286e+00 | 2.119662e-01 | 3.223396e-02 |
| 18 | 1.285714e+00 | 1.452290e-01 | 3.270001e-02 |
| 19 | 1.357143e+00 | 7.523166e-02 | 3.164376e-02 |
| 20 | 1.428571e+00 | 5.203677e-03 | 2.909967e-02 |
| 21 | 1.500000e+00 | -6.160455e-02 | 2.517885e-02 |
| 22 | 1.571429e+00 | -1.220858e-01 | 2.006329e-02 |
| 23 | 1.642857e+00 | -1.734332e-01 | 1.399655e-02 |
| 24 | 1.714286e+00 | -2.132822e-01 | 7.271367e-03 |
| 25 | 1.785714e+00 | -2.398299e-01 | 2.148680e-04 |
| 26 | 1.857143e+00 | -2.519271e-01 | 6.828000e-03 |
| 27 | 1.928571e+00 | -2.491362e-01 | 1.351188e-02 |
| 28 | 2.000000e+00 | -2.317546e-01 | 1.950842e-02 |

|  |  |  |  |
| --- | --- | --- | --- |
| n | t | y | error |
| 0 | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 |
| 1 | 3.571429e-02 | 3.854199e-03 | 1.877529e-03 |
| 2 | 7.142857e-02 | 1.137482e-02 | 3.590284e-03 |
| 3 | 1.071429e-01 | 2.233301e-02 | 5.124377e-03 |
| 4 | 1.428571e-01 | 3.646155e-02 | 6.467797e-03 |
| 5 | 1.785714e-01 | 5.345802e-02 | 7.610577e-03 |
| 6 | 2.142857e-01 | 7.298845e-02 | 8.544936e-03 |
| 7 | 2.500000e-01 | 9.469134e-02 | 9.265391e-03 |
| 8 | 2.857143e-01 | 1.181821e-01 | 9.768843e-03 |
| 9 | 3.214286e-01 | 1.430576e-01 | 1.005463e-02 |
| 10 | 3.571429e-01 | 1.689012e-01 | 1.012454e-02 |
| 11 | 3.928571e-01 | 1.952880e-01 | 9.982818e-03 |
| 12 | 4.285714e-01 | 2.217898e-01 | 9.636114e-03 |
| 13 | 4.642857e-01 | 2.479802e-01 | 9.093409e-03 |
| 14 | 5.000000e-01 | 2.734405e-01 | 8.365921e-03 |
| 15 | 5.357143e-01 | 2.977641e-01 | 7.466968e-03 |
| 16 | 5.714286e-01 | 3.205619e-01 | 6.411815e-03 |
| 17 | 6.071429e-01 | 3.414670e-01 | 5.217493e-03 |
| 18 | 6.428571e-01 | 3.601392e-01 | 3.902587e-03 |
| 19 | 6.785714e-01 | 3.762693e-01 | 2.487019e-03 |
| 20 | 7.142857e-01 | 3.895825e-01 | 9.918030e-04 |
| 21 | 7.500000e-01 | 3.998422e-01 | 5.612118e-04 |
| 22 | 7.857143e-01 | 4.068527e-01 | 2.149607e-03 |
| 23 | 8.214286e-01 | 4.104616e-01 | 3.750670e-03 |
| 24 | 8.571429e-01 | 4.105613e-01 | 5.341672e-03 |
| 25 | 8.928571e-01 | 4.070911e-01 | 6.900146e-03 |
| 26 | 9.285714e-01 | 4.000369e-01 | 8.404166e-03 |
| 27 | 9.642857e-01 | 3.894319e-01 | 9.832614e-03 |
| 28 | 1.000000e+00 | 3.753561e-01 | 1.116544e-02 |
| 29 | 1.035714e+00 | 3.579348e-01 | 1.238391e-02 |
| 30 | 1.071429e+00 | 3.373375e-01 | 1.347082e-02 |
| 31 | 1.107143e+00 | 3.137752e-01 | 1.441076e-02 |
| 32 | 1.142857e+00 | 2.874982e-01 | 1.519022e-02 |
| 33 | 1.178571e+00 | 2.587923e-01 | 1.579783e-02 |
| 34 | 1.214286e+00 | 2.279757e-01 | 1.622445e-02 |
| 35 | 1.250000e+00 | 1.953946e-01 | 1.646332e-02 |
| 36 | 1.285714e+00 | 1.614189e-01 | 1.651010e-02 |
| 37 | 1.321429e+00 | 1.264373e-01 | 1.636295e-02 |
| 38 | 1.357143e+00 | 9.085288e-02 | 1.602254e-02 |
| 39 | 1.392857e+00 | 5.507736e-02 | 1.549204e-02 |
| 40 | 1.428571e+00 | 1.952627e-02 | 1.477708e-02 |
| 41 | 1.464286e+00 | -1.538646e-02 | 1.388564e-02 |
| 42 | 1.500000e+00 | -4.925373e-02 | 1.282803e-02 |
| 43 | 1.535714e+00 | -8.168043e-02 | 1.161666e-02 |
| 44 | 1.571429e+00 | -1.122885e-01 | 1.026594e-02 |
| 45 | 1.607143e+00 | -1.407215e-01 | 8.792095e-03 |
| 46 | 1.642857e+00 | -1.666496e-01 | 7.212930e-03 |
| 47 | 1.678571e+00 | -1.897733e-01 | 5.547626e-03 |
| 48 | 1.714286e+00 | -2.098273e-01 | 3.816488e-03 |
| 49 | 1.750000e+00 | -2.265842e-01 | 2.040690e-03 |
| 50 | 1.785714e+00 | -2.398571e-01 | 2.420062e-04 |
| 51 | 1.821429e+00 | -2.495019e-01 | 1.557466e-03 |
| 52 | 1.857143e+00 | -2.554195e-01 | 3.335585e-03 |
| 53 | 1.892857e+00 | -2.575568e-01 | 5.070444e-03 |
| 54 | 1.928571e+00 | -2.559075e-01 | 6.740650e-03 |
| 55 | 1.964286e+00 | -2.505120e-01 | 8.325595e-03 |
| 56 | 2.000000e+00 | -2.414573e-01 | 9.805713e-03 |

|  |  |  |  |
| --- | --- | --- | --- |
| n | t | y | error |
| 0 | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 |
| 1 | 1.785714e-02 | 9.828299e-04 | 4.851851e-04 |
| 2 | 3.571429e-02 | 2.927309e-03 | 9.506387e-04 |
| 3 | 5.357143e-02 | 5.809562e-03 | 1.395399e-03 |
| 4 | 7.142857e-02 | 9.603092e-03 | 1.818557e-03 |
| 5 | 8.928571e-02 | 1.427885e-02 | 2.219258e-03 |
| 6 | 1.071429e-01 | 1.980534e-02 | 2.596710e-03 |
| 7 | 1.250000e-01 | 2.614869e-02 | 2.950178e-03 |
| 8 | 1.428571e-01 | 3.327275e-02 | 3.278995e-03 |
| 9 | 1.607143e-01 | 4.113923e-02 | 3.582558e-03 |
| 10 | 1.785714e-01 | 4.970778e-02 | 3.860334e-03 |
| 11 | 1.964286e-01 | 5.893614e-02 | 4.111860e-03 |
| 12 | 2.142857e-01 | 6.878026e-02 | 4.336747e-03 |
| 13 | 2.321429e-01 | 7.919441e-02 | 4.534676e-03 |
| 14 | 2.500000e-01 | 9.013135e-02 | 4.705408e-03 |
| 15 | 2.678571e-01 | 1.015425e-01 | 4.848776e-03 |
| 16 | 2.857143e-01 | 1.133779e-01 | 4.964692e-03 |
| 17 | 3.035714e-01 | 1.255867e-01 | 5.053145e-03 |
| 18 | 3.214286e-01 | 1.381171e-01 | 5.114200e-03 |
| 19 | 3.392857e-01 | 1.509164e-01 | 5.148001e-03 |
| 20 | 3.571429e-01 | 1.639314e-01 | 5.154769e-03 |
| 21 | 3.750000e-01 | 1.771085e-01 | 5.134801e-03 |
| 22 | 3.928571e-01 | 1.903937e-01 | 5.088472e-03 |
| 23 | 4.107143e-01 | 2.037329e-01 | 5.016230e-03 |
| 24 | 4.285714e-01 | 2.170723e-01 | 4.918598e-03 |
| 25 | 4.464286e-01 | 2.303579e-01 | 4.796170e-03 |
| 26 | 4.642857e-01 | 2.435364e-01 | 4.649612e-03 |
| 27 | 4.821429e-01 | 2.565551e-01 | 4.479659e-03 |
| 28 | 5.000000e-01 | 2.693617e-01 | 4.287110e-03 |
| 29 | 5.178571e-01 | 2.819050e-01 | 4.072832e-03 |
| 30 | 5.357143e-01 | 2.941349e-01 | 3.837752e-03 |
| 31 | 5.535714e-01 | 3.060022e-01 | 3.582854e-03 |
| 32 | 5.714286e-01 | 3.174592e-01 | 3.309181e-03 |
| 33 | 5.892857e-01 | 3.284598e-01 | 3.017827e-03 |
| 34 | 6.071429e-01 | 3.389594e-01 | 2.709937e-03 |
| 35 | 6.250000e-01 | 3.489151e-01 | 2.386700e-03 |
| 36 | 6.428571e-01 | 3.582860e-01 | 2.049350e-03 |
| 37 | 6.607143e-01 | 3.670331e-01 | 1.699158e-03 |
| 38 | 6.785714e-01 | 3.751197e-01 | 1.337430e-03 |
| 39 | 6.964286e-01 | 3.825112e-01 | 9.655039e-04 |
| 40 | 7.142857e-01 | 3.891754e-01 | 5.847437e-04 |
| 41 | 7.321429e-01 | 3.950826e-01 | 1.965362e-04 |
| 42 | 7.500000e-01 | 4.002057e-01 | 1.977133e-04 |
| 43 | 7.678571e-01 | 4.045200e-01 | 5.965854e-04 |
| 44 | 7.857143e-01 | 4.080037e-01 | 9.986510e-04 |
| 45 | 8.035714e-01 | 4.106376e-01 | 1.402475e-03 |
| 46 | 8.214286e-01 | 4.124056e-01 | 1.806623e-03 |
| 47 | 8.392857e-01 | 4.132942e-01 | 2.209662e-03 |
| 48 | 8.571429e-01 | 4.132929e-01 | 2.610167e-03 |
| 49 | 8.750000e-01 | 4.123941e-01 | 3.006728e-03 |
| 50 | 8.928571e-01 | 4.105933e-01 | 3.397947e-03 |
| 51 | 9.107143e-01 | 4.078889e-01 | 3.782450e-03 |
| 52 | 9.285714e-01 | 4.042822e-01 | 4.158887e-03 |
| 53 | 9.464286e-01 | 3.997775e-01 | 4.525939e-03 |
| 54 | 9.642857e-01 | 3.943822e-01 | 4.882316e-03 |
| 55 | 9.821429e-01 | 3.881065e-01 | 5.226770e-03 |
| 56 | 1.000000e+00 | 3.809634e-01 | 5.558091e-03 |
| 57 | 1.017857e+00 | 3.729690e-01 | 5.875115e-03 |
| 58 | 1.035714e+00 | 3.641420e-01 | 6.176726e-03 |
| 59 | 1.053571e+00 | 3.545038e-01 | 6.461860e-03 |
| 60 | 1.071429e+00 | 3.440788e-01 | 6.729507e-03 |
| 61 | 1.089286e+00 | 3.328935e-01 | 6.978718e-03 |
| 62 | 1.107143e+00 | 3.209774e-01 | 7.208601e-03 |
| 63 | 1.125000e+00 | 3.083620e-01 | 7.418332e-03 |
| 64 | 1.142857e+00 | 2.950813e-01 | 7.607149e-03 |
| 65 | 1.160714e+00 | 2.811715e-01 | 7.774362e-03 |
| 66 | 1.178571e+00 | 2.666708e-01 | 7.919351e-03 |
| 67 | 1.196429e+00 | 2.516195e-01 | 8.041568e-03 |
| 68 | 1.214286e+00 | 2.360597e-01 | 8.140538e-03 |
| 69 | 1.232143e+00 | 2.200350e-01 | 8.215865e-03 |
| 70 | 1.250000e+00 | 2.035907e-01 | 8.267228e-03 |
| 71 | 1.267857e+00 | 1.867737e-01 | 8.294383e-03 |
| 72 | 1.285714e+00 | 1.696318e-01 | 8.297165e-03 |
| 73 | 1.303571e+00 | 1.522142e-01 | 8.275490e-03 |
| 74 | 1.321429e+00 | 1.345709e-01 | 8.229350e-03 |
| 75 | 1.339286e+00 | 1.167528e-01 | 8.158819e-03 |
| 76 | 1.357143e+00 | 9.881137e-02 | 8.064048e-03 |
| 77 | 1.375000e+00 | 8.079855e-02 | 7.945267e-03 |
| 78 | 1.392857e+00 | 6.276662e-02 | 7.802785e-03 |
| 79 | 1.410714e+00 | 4.476798e-02 | 7.636984e-03 |
| 80 | 1.428571e+00 | 2.685502e-02 | 7.448325e-03 |
| 81 | 1.446429e+00 | 9.079933e-03 | 7.237340e-03 |
| 82 | 1.464286e+00 | -8.505451e-03 | 7.004635e-03 |
| 83 | 1.482143e+00 | -2.584982e-02 | 6.750886e-03 |
| 84 | 1.500000e+00 | -4.290254e-02 | 6.476836e-03 |
| 85 | 1.517857e+00 | -5.961383e-02 | 6.183294e-03 |
| 86 | 1.535714e+00 | -7.593491e-02 | 5.871131e-03 |
| 87 | 1.553571e+00 | -9.181816e-02 | 5.541281e-03 |
| 88 | 1.571429e+00 | -1.072173e-01 | 5.194731e-03 |
| 89 | 1.589286e+00 | -1.220874e-01 | 4.832527e-03 |
| 90 | 1.607143e+00 | -1.363852e-01 | 4.455761e-03 |
| 91 | 1.625000e+00 | -1.500692e-01 | 4.065576e-03 |
| 92 | 1.642857e+00 | -1.630999e-01 | 3.663156e-03 |
| 93 | 1.660714e+00 | -1.754394e-01 | 3.249726e-03 |
| 94 | 1.678571e+00 | -1.870522e-01 | 2.826547e-03 |
| 95 | 1.696429e+00 | -1.979051e-01 | 2.394913e-03 |
| 96 | 1.714286e+00 | -2.079670e-01 | 1.956143e-03 |
| 97 | 1.732143e+00 | -2.172093e-01 | 1.511583e-03 |
| 98 | 1.750000e+00 | -2.256061e-01 | 1.062596e-03 |
| 99 | 1.767857e+00 | -2.331339e-01 | 6.105606e-04 |
| 100 | 1.785714e+00 | -2.397719e-01 | 1.568670e-04 |
| 101 | 1.803571e+00 | -2.455022e-01 | 2.970889e-04 |
| 102 | 1.821429e+00 | -2.503095e-01 | 7.499093e-04 |
| 103 | 1.839286e+00 | -2.541814e-01 | 1.200199e-03 |
| 104 | 1.857143e+00 | -2.571085e-01 | 1.646569e-03 |
| 105 | 1.875000e+00 | -2.590843e-01 | 2.087643e-03 |
| 106 | 1.892857e+00 | -2.601052e-01 | 2.522059e-03 |
| 107 | 1.910714e+00 | -2.601705e-01 | 2.948476e-03 |
| 108 | 1.928571e+00 | -2.592825e-01 | 3.365576e-03 |
| 109 | 1.946429e+00 | -2.574466e-01 | 3.772072e-03 |
| 110 | 1.964286e+00 | -2.546709e-01 | 4.166707e-03 |
| 111 | 1.982143e+00 | -2.509665e-01 | 4.548261e-03 |
| 112 | 2.000000e+00 | -2.463475e-01 | 4.915554e-03 |

## Wykres funkcji

Obraz zawierający Wykres, linia, diagram, tekst

Opis wygenerowany automatycznie

## Wykres zależności od

Obraz zawierający tekst, linia, diagram, Wykres

Opis wygenerowany automatycznie

Współczynnik nachylenia wykresu wynosi i jest to rząd dokładności metody pośredniej Eulera. Współczynnik ten koreluje z rzędem dokładności (wykładnikami przy wartości ) dla tej metody, gdzie lokalny błąd wynosi .

# Zadanie 3

## Metoda pośrednia Eulera w języku Python, używając biblioteki *numpy*

1. def exact\_solution\_getter(q: float, r: float, s: float) -> ExactSolutionType:

2. def exact\_solution(x: float) -> float:

3. negative\_half\_q: float = -q / 2

4. square\_root: float = sqrt(square(q) - 4 \* r)

5. lambda\_1: float = negative\_half\_q - square\_root / 2

6. lambda\_2: float = negative\_half\_q + square\_root / 2

7. A: float = (exp(5 \* lambda\_2) - 1) / (exp(5 \* lambda\_1) - exp(5 \* lambda\_2))

8. B: float = (exp(5 \* lambda\_1) - 1) / (exp(5 \* lambda\_2) - exp(5 \* lambda\_1))

9. return s / r \* (A \* exp(lambda\_1 \* x) + B \* exp(lambda\_2 \* x) + 1)

10. return exact\_solution

11.

12. def lower\_upper\_decomposition(

13. A: FloatArray2D,

14. ) -> tuple[FloatArray2D, FloatArray2D]:

15. n: int = A.shape[0]

16. a: FloatArray2D = A.copy()

17. for k in range(n - 1):

18. akk: float = a[k][k]

19. for i in range(k + 1, n):

20. aux: float = a[i][k] / akk if akk else 0

21. for j in range(k + 1, n):

22. a[i][j] -= a[k][j] \* aux

23. a[i][k] = aux

24. U: FloatArray2D = triu(a)

25. L: FloatArray2D = a - U + eye(n)

26. return L, U

27.

28. def eliminate\_forward(L: FloatArray2D, B: FloatArray) -> FloatArray:

29. n: int = L.shape[0]

30. b: FloatArray = B.copy()

31. for k in range(n - 1):

32. for i in range(k + 1, n):

33. b[i] -= b[k] \* L[i][k]

34. return b

35.

36. def substitute\_backward(U: FloatArray2D, Y: FloatArray) -> FloatArray:

37. n: int = U.shape[0]

38. y: FloatArray = Y.copy()

39. y[n-1] /= U[n-1][n-1]

40. for i in range(n-2, -1, -1):

41. s: float = 0.0

42. for j in range(i+1, n):

43. s += U[i][j] \* y[j]

44. y[i] -= s

45. y[i] /= U[i][i]

46. return y

47.

48. def solve\_2\_degree\_differential\_equation\_getter(

49. q: float, r: float, s: float, x\_min: float, x\_max: float,

50. y\_for\_x\_min: float, y\_for\_x\_max: float, exact\_solution: ExactSolutionType,

51. ) -> Solve2DegreeDifferentialEquationType:

52. def solve\_2\_degree\_differential\_equation(

53. N: int,

54. ) -> Solve2DegreeDifferentialEquationReturnType:

55. h: float = (x\_max - x\_min) / N

56.

57. A: FloatArray2D = zeros((N + 1, N + 1))

58. A[0][0] = 1

59. A[N][N] = 1

60.

61. B: FloatArray = repeat(s, N + 1)

62. B[0] = y\_for\_x\_min

63. B[N] = y\_for\_x\_max

64.

65. for i in arange(1, N):

66. A[i][i - 1] = 1 / square(h) - q / 2 / h

67. A[i][i] = r - 2 / square(h)

68. A[i][i + 1] = 1 / square(h) + q / 2 / h

69.

70. L, U = lower\_upper\_decomposition(A)

71. eliminated: FloatArray = eliminate\_forward(L, B)

72. ys: FloatArray = substitute\_backward(U, eliminated)

73.

74. xs: list[float] = []

75. errors = []

76. for i in arange(0, N + 1):

77. xs.append(x\_min + h \* i)

78. errors.append(abs(exact\_solution(xs[i]) - ys[i]))

79.

80. return array(xs), ys, array(errors), h

81. return solve\_2\_degree\_differential\_equation

## Problem początkowy

## Dane i ilość iteracji N

## Ścisłe rozwiązanie

gdzie:

### Sprawdzenie rozwiązania dla warunków brzegowych

Ścisłe rozwiązanie spełnia zadane równanie różniczkowe dla warunków brzegowych. Obliczone i  nie są dokładnie równe ze względu na błąd zaokrąglenia.

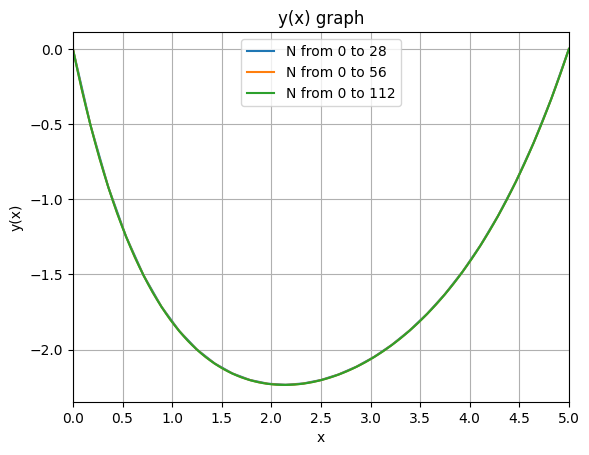
## Kolejne wartości obliczone za pomocą dekompozycji LU

|  |  |  |  |
| --- | --- | --- | --- |
| n | x | y | error |
| 0 | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 |
| 1 | 1.785714e-01 | -5.100519e-01 | 3.153510e-04 |
| 2 | 3.571429e-01 | -9.196016e-01 | 5.373573e-04 |
| 3 | 5.357143e-01 | -1.247273e+00 | 6.922712e-04 |
| 4 | 7.142857e-01 | -1.508083e+00 | 7.998338e-04 |
| 5 | 8.928571e-01 | -1.714124e+00 | 8.747893e-04 |
| 6 | 1.071429e+00 | -1.875103e+00 | 9.280526e-04 |
| 7 | 1.250000e+00 | -1.998789e+00 | 9.676077e-04 |
| 8 | 1.428571e+00 | -2.091364e+00 | 9.991941e-04 |
| 9 | 1.607143e+00 | -2.157715e+00 | 1.026829e-03 |
| 10 | 1.785714e+00 | -2.201659e+00 | 1.053202e-03 |
| 11 | 1.964286e+00 | -2.226131e+00 | 1.079963e-03 |
| 12 | 2.142857e+00 | -2.233329e+00 | 1.107937e-03 |
| 13 | 2.321429e+00 | -2.224828e+00 | 1.137273e-03 |
| 14 | 2.500000e+00 | -2.201676e+00 | 1.167539e-03 |
| 15 | 2.678571e+00 | -2.164456e+00 | 1.197782e-03 |
| 16 | 2.857143e+00 | -2.113346e+00 | 1.226547e-03 |
| 17 | 3.035714e+00 | -2.048150e+00 | 1.251871e-03 |
| 18 | 3.214286e+00 | -1.968327e+00 | 1.271245e-03 |
| 19 | 3.392857e+00 | -1.873003e+00 | 1.281557e-03 |
| 20 | 3.571429e+00 | -1.760976e+00 | 1.279009e-03 |
| 21 | 3.750000e+00 | -1.630711e+00 | 1.259004e-03 |
| 22 | 3.928571e+00 | -1.480329e+00 | 1.216015e-03 |
| 23 | 4.107143e+00 | -1.307585e+00 | 1.143417e-03 |
| 24 | 4.285714e+00 | -1.109839e+00 | 1.033297e-03 |
| 25 | 4.464286e+00 | -8.840211e-01 | 8.762183e-04 |
| 26 | 4.642857e+00 | -6.265878e-01 | 6.609511e-04 |
| 27 | 4.821429e+00 | -3.334676e-01 | 3.741537e-04 |
| 28 | 5.000000e+00 | 0.000000e+00 | 3.132415e-16 |

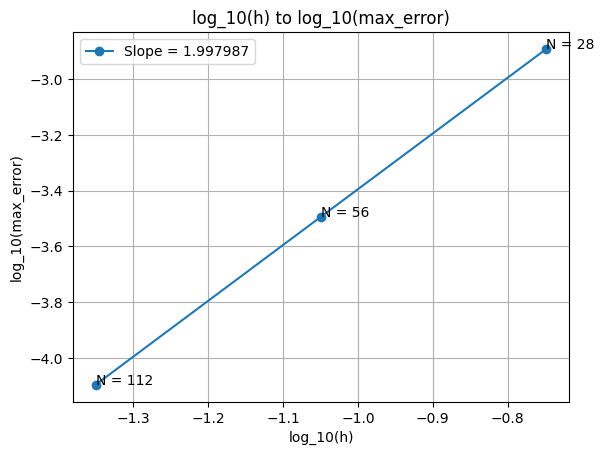
|  |  |  |  |
| --- | --- | --- | --- |
| n | t | y | error |
| 0 | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 |
| 1 | 8.928571e-02 | -2.690435e-01 | 4.294541e-05 |
| 2 | 1.785714e-01 | -5.102882e-01 | 7.906035e-05 |
| 3 | 2.678571e-01 | -7.264590e-01 | 1.093539e-04 |
| 4 | 3.571429e-01 | -9.200042e-01 | 1.347035e-04 |
| 5 | 4.464286e-01 | -1.093123e+00 | 1.558709e-04 |
| 6 | 5.357143e-01 | -1.247791e+00 | 1.735168e-04 |
| 7 | 6.250000e-01 | -1.385781e+00 | 1.882130e-04 |
| 8 | 7.142857e-01 | -1.508683e+00 | 2.004537e-04 |
| 9 | 8.035714e-01 | -1.617923e+00 | 2.106649e-04 |
| 10 | 8.928571e-01 | -1.714780e+00 | 2.192133e-04 |
| 11 | 9.821429e-01 | -1.800397e+00 | 2.264134e-04 |
| 12 | 1.071429e+00 | -1.875799e+00 | 2.325340e-04 |
| 13 | 1.160714e+00 | -1.941899e+00 | 2.378044e-04 |
| 14 | 1.250000e+00 | -1.999514e+00 | 2.424188e-04 |
| 15 | 1.339286e+00 | -2.049370e+00 | 2.465407e-04 |
| 16 | 1.428571e+00 | -2.092113e+00 | 2.503072e-04 |
| 17 | 1.517857e+00 | -2.128315e+00 | 2.538318e-04 |
| 18 | 1.607143e+00 | -2.158484e+00 | 2.572072e-04 |
| 19 | 1.696429e+00 | -2.183065e+00 | 2.605081e-04 |
| 20 | 1.785714e+00 | -2.202448e+00 | 2.637931e-04 |
| 21 | 1.875000e+00 | -2.216975e+00 | 2.671062e-04 |
| 22 | 1.964286e+00 | -2.226940e+00 | 2.704789e-04 |
| 23 | 2.053571e+00 | -2.232597e+00 | 2.739309e-04 |
| 24 | 2.142857e+00 | -2.234159e+00 | 2.774717e-04 |
| 25 | 2.232143e+00 | -2.231805e+00 | 2.811009e-04 |
| 26 | 2.321429e+00 | -2.225681e+00 | 2.848090e-04 |
| 27 | 2.410714e+00 | -2.215901e+00 | 2.885783e-04 |
| 28 | 2.500000e+00 | -2.202551e+00 | 2.923828e-04 |
| 29 | 2.589286e+00 | -2.185691e+00 | 2.961887e-04 |
| 30 | 2.678571e+00 | -2.165354e+00 | 2.999543e-04 |
| 31 | 2.767857e+00 | -2.141550e+00 | 3.036302e-04 |
| 32 | 2.857143e+00 | -2.114265e+00 | 3.071590e-04 |
| 33 | 2.946429e+00 | -2.083464e+00 | 3.104753e-04 |
| 34 | 3.035714e+00 | -2.049088e+00 | 3.135049e-04 |
| 35 | 3.125000e+00 | -2.011060e+00 | 3.161651e-04 |
| 36 | 3.214286e+00 | -1.969280e+00 | 3.183636e-04 |
| 37 | 3.303571e+00 | -1.923628e+00 | 3.199981e-04 |
| 38 | 3.392857e+00 | -1.873963e+00 | 3.209554e-04 |
| 39 | 3.482143e+00 | -1.820126e+00 | 3.211111e-04 |
| 40 | 3.571429e+00 | -1.761934e+00 | 3.203282e-04 |
| 41 | 3.660714e+00 | -1.699185e+00 | 3.184563e-04 |
| 42 | 3.750000e+00 | -1.631655e+00 | 3.153302e-04 |
| 43 | 3.839286e+00 | -1.559096e+00 | 3.107694e-04 |
| 44 | 3.928571e+00 | -1.481241e+00 | 3.045760e-04 |
| 45 | 4.017857e+00 | -1.397795e+00 | 2.965335e-04 |
| 46 | 4.107143e+00 | -1.308442e+00 | 2.864054e-04 |
| 47 | 4.196429e+00 | -1.212838e+00 | 2.739332e-04 |
| 48 | 4.285714e+00 | -1.110613e+00 | 2.588346e-04 |
| 49 | 4.375000e+00 | -1.001369e+00 | 2.408017e-04 |
| 50 | 4.464286e+00 | -8.846779e-01 | 2.194982e-04 |
| 51 | 4.553571e+00 | -7.600799e-01 | 1.945578e-04 |
| 52 | 4.642857e+00 | -6.270832e-01 | 1.655808e-04 |
| 53 | 4.732143e+00 | -4.851605e-01 | 1.321321e-04 |
| 54 | 4.821429e+00 | -3.337481e-01 | 9.373747e-05 |
| 55 | 4.910714e+00 | -1.722427e-01 | 4.988071e-05 |
| 56 | 5.000000e+00 | 0.000000e+00 | 3.132415e-16 |

|  |  |  |  |
| --- | --- | --- | --- |
| n | t | y | error |
| 0 | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 |
| 1 | 4.464286e-02 | -1.381953e-01 | 5.602870e-06 |
| 2 | 8.928571e-02 | -2.690757e-01 | 1.074408e-05 |
| 3 | 1.339286e-01 | -3.930099e-01 | 1.545859e-05 |
| 4 | 1.785714e-01 | -5.103475e-01 | 1.977904e-05 |
| 5 | 2.232143e-01 | -6.214198e-01 | 2.373585e-05 |
| 6 | 2.678571e-01 | -7.265410e-01 | 2.735739e-05 |
| 7 | 3.125000e-01 | -8.260087e-01 | 3.067013e-05 |
| 8 | 3.571429e-01 | -9.201052e-01 | 3.369869e-05 |
| 9 | 4.017857e-01 | -1.009098e+00 | 3.646605e-05 |
| 10 | 4.464286e-01 | -1.093240e+00 | 3.899358e-05 |
| 11 | 4.910714e-01 | -1.172772e+00 | 4.130117e-05 |
| 12 | 5.357143e-01 | -1.247921e+00 | 4.340734e-05 |
| 13 | 5.803571e-01 | -1.318903e+00 | 4.532931e-05 |
| 14 | 6.250000e-01 | -1.385922e+00 | 4.708308e-05 |
| 15 | 6.696429e-01 | -1.449171e+00 | 4.868353e-05 |
| 16 | 7.142857e-01 | -1.508833e+00 | 5.014445e-05 |
| 17 | 7.589286e-01 | -1.565082e+00 | 5.147867e-05 |
| 18 | 8.035714e-01 | -1.618081e+00 | 5.269807e-05 |
| 19 | 8.482143e-01 | -1.667986e+00 | 5.381366e-05 |
| 20 | 8.928571e-01 | -1.714944e+00 | 5.483566e-05 |
| 21 | 9.375000e-01 | -1.759094e+00 | 5.577350e-05 |
| 22 | 9.821429e-01 | -1.800567e+00 | 5.663592e-05 |
| 23 | 1.026786e+00 | -1.839488e+00 | 5.743099e-05 |
| 24 | 1.071429e+00 | -1.875973e+00 | 5.816615e-05 |
| 25 | 1.116071e+00 | -1.910135e+00 | 5.884827e-05 |
| 26 | 1.160714e+00 | -1.942077e+00 | 5.948367e-05 |
| 27 | 1.205357e+00 | -1.971900e+00 | 6.007816e-05 |
| 28 | 1.250000e+00 | -1.999696e+00 | 6.063708e-05 |
| 29 | 1.294643e+00 | -2.025553e+00 | 6.116531e-05 |
| 30 | 1.339286e+00 | -2.049555e+00 | 6.166733e-05 |
| 31 | 1.383929e+00 | -2.071779e+00 | 6.214722e-05 |
| 32 | 1.428571e+00 | -2.092300e+00 | 6.260869e-05 |
| 33 | 1.473214e+00 | -2.111188e+00 | 6.305511e-05 |
| 34 | 1.517857e+00 | -2.128506e+00 | 6.348954e-05 |
| 35 | 1.562500e+00 | -2.144316e+00 | 6.391473e-05 |
| 36 | 1.607143e+00 | -2.158677e+00 | 6.433313e-05 |
| 37 | 1.651786e+00 | -2.171641e+00 | 6.474694e-05 |
| 38 | 1.696429e+00 | -2.183260e+00 | 6.515811e-05 |
| 39 | 1.741071e+00 | -2.193580e+00 | 6.556834e-05 |
| 40 | 1.785714e+00 | -2.202646e+00 | 6.597913e-05 |
| 41 | 1.830357e+00 | -2.210498e+00 | 6.639174e-05 |
| 42 | 1.875000e+00 | -2.217175e+00 | 6.680724e-05 |
| 43 | 1.919643e+00 | -2.222713e+00 | 6.722653e-05 |
| 44 | 1.964286e+00 | -2.227143e+00 | 6.765031e-05 |
| 45 | 2.008929e+00 | -2.230497e+00 | 6.807910e-05 |
| 46 | 2.053571e+00 | -2.232802e+00 | 6.851327e-05 |
| 47 | 2.098214e+00 | -2.234085e+00 | 6.895305e-05 |
| 48 | 2.142857e+00 | -2.234367e+00 | 6.939848e-05 |
| 49 | 2.187500e+00 | -2.233671e+00 | 6.984948e-05 |
| 50 | 2.232143e+00 | -2.232016e+00 | 7.030583e-05 |
| 51 | 2.276786e+00 | -2.229419e+00 | 7.076717e-05 |
| 52 | 2.321429e+00 | -2.225894e+00 | 7.123300e-05 |
| 53 | 2.366071e+00 | -2.221457e+00 | 7.170270e-05 |
| 54 | 2.410714e+00 | -2.216117e+00 | 7.217552e-05 |
| 55 | 2.455357e+00 | -2.209886e+00 | 7.265059e-05 |
| 56 | 2.500000e+00 | -2.202770e+00 | 7.312691e-05 |
| 57 | 2.544643e+00 | -2.194778e+00 | 7.360336e-05 |
| 58 | 2.589286e+00 | -2.185913e+00 | 7.407869e-05 |
| 59 | 2.633929e+00 | -2.176180e+00 | 7.455155e-05 |
| 60 | 2.678571e+00 | -2.165579e+00 | 7.502046e-05 |
| 61 | 2.723214e+00 | -2.154112e+00 | 7.548379e-05 |
| 62 | 2.767857e+00 | -2.141778e+00 | 7.593984e-05 |
| 63 | 2.812500e+00 | -2.128574e+00 | 7.638672e-05 |
| 64 | 2.857143e+00 | -2.114496e+00 | 7.682248e-05 |
| 65 | 2.901786e+00 | -2.099539e+00 | 7.724499e-05 |
| 66 | 2.946429e+00 | -2.083697e+00 | 7.765200e-05 |
| 67 | 2.991071e+00 | -2.066961e+00 | 7.804115e-05 |
| 68 | 3.035714e+00 | -2.049324e+00 | 7.840990e-05 |
| 69 | 3.080357e+00 | -2.030773e+00 | 7.875560e-05 |
| 70 | 3.125000e+00 | -2.011297e+00 | 7.907544e-05 |
| 71 | 3.169643e+00 | -1.990884e+00 | 7.936645e-05 |
| 72 | 3.214286e+00 | -1.969518e+00 | 7.962553e-05 |
| 73 | 3.258929e+00 | -1.947185e+00 | 7.984939e-05 |
| 74 | 3.303571e+00 | -1.923868e+00 | 8.003459e-05 |
| 75 | 3.348214e+00 | -1.899547e+00 | 8.017751e-05 |
| 76 | 3.392857e+00 | -1.874204e+00 | 8.027434e-05 |
| 77 | 3.437500e+00 | -1.847818e+00 | 8.032111e-05 |
| 78 | 3.482143e+00 | -1.820367e+00 | 8.031362e-05 |
| 79 | 3.526786e+00 | -1.791827e+00 | 8.024750e-05 |
| 80 | 3.571429e+00 | -1.762174e+00 | 8.011816e-05 |
| 81 | 3.616071e+00 | -1.731382e+00 | 7.992079e-05 |
| 82 | 3.660714e+00 | -1.699424e+00 | 7.965034e-05 |
| 83 | 3.705357e+00 | -1.666270e+00 | 7.930154e-05 |
| 84 | 3.750000e+00 | -1.631891e+00 | 7.886887e-05 |
| 85 | 3.794643e+00 | -1.596255e+00 | 7.834656e-05 |
| 86 | 3.839286e+00 | -1.559329e+00 | 7.772855e-05 |
| 87 | 3.883929e+00 | -1.521079e+00 | 7.700853e-05 |
| 88 | 3.928571e+00 | -1.481469e+00 | 7.617989e-05 |
| 89 | 3.973214e+00 | -1.440462e+00 | 7.523570e-05 |
| 90 | 4.017857e+00 | -1.398018e+00 | 7.416873e-05 |
| 91 | 4.062500e+00 | -1.354097e+00 | 7.297144e-05 |
| 92 | 4.107143e+00 | -1.308657e+00 | 7.163591e-05 |
| 93 | 4.151786e+00 | -1.261654e+00 | 7.015389e-05 |
| 94 | 4.196429e+00 | -1.213044e+00 | 6.851676e-05 |
| 95 | 4.241071e+00 | -1.162778e+00 | 6.671548e-05 |
| 96 | 4.285714e+00 | -1.110807e+00 | 6.474066e-05 |
| 97 | 4.330357e+00 | -1.057082e+00 | 6.258244e-05 |
| 98 | 4.375000e+00 | -1.001550e+00 | 6.023056e-05 |
| 99 | 4.419643e+00 | -9.441553e-01 | 5.767427e-05 |
| 100 | 4.464286e+00 | -8.848425e-01 | 5.490237e-05 |
| 101 | 4.508929e+00 | -8.235528e-01 | 5.190315e-05 |
| 102 | 4.553571e+00 | -7.602258e-01 | 4.866440e-05 |
| 103 | 4.598214e+00 | -6.947989e-01 | 4.517336e-05 |
| 104 | 4.642857e+00 | -6.272073e-01 | 4.141672e-05 |
| 105 | 4.687500e+00 | -5.573840e-01 | 3.738057e-05 |
| 106 | 4.732143e+00 | -4.852596e-01 | 3.305041e-05 |
| 107 | 4.776786e+00 | -4.107625e-01 | 2.841110e-05 |
| 108 | 4.821429e+00 | -3.338184e-01 | 2.344685e-05 |
| 109 | 4.866071e+00 | -2.543507e-01 | 1.814118e-05 |
| 110 | 4.910714e+00 | -1.722801e-01 | 1.247690e-05 |
| 111 | 4.955357e+00 | -8.752477e-02 | 6.436077e-06 |
| 112 | 5.000000e+00 | 0.000000e+00 | 3.132415e-16 |

## Wykres funkcji



## Wykres zależności od



Współczynnik nachylenia wykresu wynosi i jest to rząd dokładności rozwiązania równania różniczkowego 2-go rzędu. Współczynnik ten koreluje z rzędem dokładności (wykładnikami przy wartości ) obliczania 1. i 2. pochodnej za pomocą różnicy centralnej – zostały one użyte do wypełnienia macierzy , gdzie lokalny błąd wynosi .

# Appendix

## ex\_1.py

1. from matplotlib.pyplot import figure, Axes, Figure

2. from numpy import float64, cos, sin, abs, log10, array

3. from numpy.typing import NDArray

4. from pandas import DataFrame, set\_option, reset\_option

5. from typing import TypeAlias, Callable

6.

7. FloatArray: TypeAlias = NDArray[float64]

8. Function: TypeAlias = Callable[[float], float]

9. DerivativeMethod: TypeAlias = Callable[[Function, float, float], float]

10.

11. set\_option('display.float\_format', lambda x: '%.6e' % x)

12.

13. def progresive\_diff\_derivative(f: Function, x: float, h: float) -> float:

14. return (f(x + h) - f(x)) / h

15.

16. def regresive\_diff\_derivative(f: Function, x: float, h: float) -> float:

17. return (f(x) - f(x - h)) / h

18.

19. def central\_diff\_derivative(f: Function, x: float, h: float) -> float:

20. return (f(x + h) - f(x - h)) / (2 \* h)

21.

22. def central\_diff\_derivative\_2(f: Function, x: float, h: float) -> float:

23. return (f(x - h) - 2 \* f(x) + f(x + h)) / h \*\* 2

24.

25. def calculate\_slope(xs: FloatArray, ys: FloatArray) -> float:

26. return (ys[-1] - ys[0]) / (xs[-1] - xs[0])

27.

28. def process(diff\_derivative\_name: str,

29. diff\_derivative\_method: DerivativeMethod,

30. f: Function,

31. x: float,

32. hs: FloatArray,

33. exact\_value: float,

34. ) -> FloatArray:

35. diff\_derivatives: list[float] = []

36. diff\_derivative\_errors: list[float] = []

37. degree: str = '2' if diff\_derivative\_name.endswith('2') else ''

38. print(f'f\_derivative{degree} = {exact\_value:.6e}')

39. print(diff\_derivative\_name)

40. for h in hs:

41. diff\_derivative: float = diff\_derivative\_method(f, x, h)

42. diff\_derivatives.append(diff\_derivative)

43. diff\_derivative\_error: float = abs(exact\_value - diff\_derivative)

44. diff\_derivative\_errors.append(diff\_derivative\_error)

45. data: dict[str, list[float]] = {

46. 'values': diff\_derivatives,

47. 'errors': diff\_derivative\_errors

48. }

49. data\_frame = DataFrame(data, index = hs).rename\_axis('h', axis=1)

50. print(data\_frame, '\n')

51. return log10(diff\_derivative\_errors)

52.

53. if \_\_name\_\_ == '\_\_main\_\_':

54. x: float = 4.63

55. hs: FloatArray = array([0.1, 0.05, 0.025, 0.0125])

56.

57. f: Function = lambda x: cos(x)

58. f\_p: Function = lambda x: -sin(x)

59. f\_pp: Function = lambda x: -cos(x)

60.

61. f\_derivative: float = f\_p(x)

62. f\_derivative2: float = f\_pp(x)

63.

64. log10\_regr\_diff\_errors: FloatArray = process(

65. 'regresive\_diff\_derivative',

66. regresive\_diff\_derivative,

67. f, x, hs, f\_derivative,

68. )

69. log10\_cent\_diff\_errors: FloatArray = process(

70. 'central\_diff\_derivative',

71. central\_diff\_derivative,

72. f, x, hs, f\_derivative,

73. )

74. log10\_cent\_diff2\_errors: FloatArray = process(

75. 'central\_diff\_derivative\_2',

76. central\_diff\_derivative\_2,

77. f, x, hs, f\_derivative2,

78. )

79.

80. log10\_hs: FloatArray = log10(hs)

81. regr\_diff\_slope: float = calculate\_slope(log10\_hs, log10\_regr\_diff\_errors)

82. cent\_diff\_slope: float = calculate\_slope(log10\_hs, log10\_cent\_diff\_errors)

83. cent\_diff2\_slope: float = calculate\_slope(log10\_hs, log10\_cent\_diff2\_errors)

84. regr\_diff\_slope\_text: str = f'slope = {regr\_diff\_slope:.6f}'

85. cent\_diff\_slope\_text: str = f'slope = {cent\_diff\_slope:.6f}'

86. cent\_diff2\_slope\_text: str = f'slope = {cent\_diff2\_slope:.6f}'

87. regr\_diff\_label = f'Regresive 1st derivative, ' + regr\_diff\_slope\_text

88. cent\_diff\_label = f'Central 1st derivative, ' + cent\_diff\_slope\_text

89. cent\_diff2\_label = f'Central 2nd derivative, ' + cent\_diff2\_slope\_text

90.

91. axes: Axes = figure().add\_subplot()

92. axes.set\_xlabel('log\_10(h)')

93. axes.set\_ylabel('log\_10(|diff\_derivative - exact\_derivative|)')

94. axes.plot(log10\_hs, log10\_regr\_diff\_errors, label = regr\_diff\_label)

95. axes.plot(log10\_hs, log10\_cent\_diff\_errors, label = cent\_diff\_label)

96. axes.plot(log10\_hs, log10\_cent\_diff2\_errors, label = cent\_diff2\_label)

97. axes.grid()

98. axes.legend()

99. if isinstance(fig := axes.get\_figure(), Figure):

100. fig.show()

101.

102. reset\_option('display.float\_format')

## ex\_2.py

1. from matplotlib.pyplot import figure, Axes, Figure

2. from numpy import (sin, cos, exp, square, pi, int\_, float64, arange, abs, log10,

3. array, append)

4. from numpy.typing import NDArray

5. from pandas import DataFrame, set\_option, reset\_option

6. from typing import TypeAlias, Callable, Annotated, Literal

7.

8. FloatArray: TypeAlias = Annotated[NDArray[float64], Literal[1]]

9. FloatArray2D: TypeAlias = Annotated[NDArray[float64], Literal[2]]

10. TCurrentGetterType: TypeAlias = Callable[[int], float]

11. YNextGetterType: TypeAlias = Callable[[float, float], float]

12. ExactSolutionType: TypeAlias = Callable[[float], float]

13. ImplicitEulerReturnType: TypeAlias = tuple[FloatArray, FloatArray, FloatArray, float]

14. ImplicitEulerType: TypeAlias = Callable[[int], ImplicitEulerReturnType]

15.

16. set\_option('display.float\_format', lambda x: '%.6e' % x)

17. set\_option('display.max\_columns', 1000)

18.

19. def y\_next\_getter\_getter(a: float, dt: float) -> YNextGetterType:

20. def y\_next\_getter(y\_previous: float, t: float) -> float:

21. return (y\_previous + dt \* sin(pi \* t)) / (1 + a \* dt)

22. return y\_next\_getter

23.

24. def exact\_solution\_getter(a: float) -> ExactSolutionType:

25. def exact\_solution(t: float) -> float:

26. numerator: float = pi \* exp(-a\*t) - pi \* cos(pi\*t) + a \* sin(pi\*t)

27. denominator: float = square(pi) + square(a)

28. return numerator / denominator

29. return exact\_solution

30.

31. def implicit\_euler\_getter(

32. a: float, t\_min: float, t\_max: float,

33. y0: float, exact\_solution: ExactSolutionType,

34. ) -> ImplicitEulerType:

35. def implicit\_euler(N: int) -> ImplicitEulerReturnType:

36. dt: float = (t\_max - t\_min) / N

37. y\_next\_getter: YNextGetterType = y\_next\_getter\_getter(a, dt)

38. current\_t\_getter: TCurrentGetterType = lambda i: t\_min + dt \* i

39. ts: list[float] = [current\_t\_getter(0)]

40. ys: list[float] = [y0]

41. errors: list[float] = [abs(exact\_solution(ts[0]) - ys[0])]

42. for i in arange(1, N + 1):

43. ts.append(current\_t\_getter(i))

44. ys.append(y\_next\_getter(ys[i - 1], ts[i]))

45. errors.append(abs(exact\_solution(ts[i]) - ys[i]))

46. return array(ts), array(ys), array(errors), dt

47. return implicit\_euler

48.

49. def get\_axes(

50. title: str, x\_limits: list[float] | None,

51. x\_ticks: FloatArray | None, x\_label: str, y\_label: str,

52. ) -> Axes:

53. axes: Axes = figure().add\_subplot()

54. axes.set\_title(title)

55. if x\_limits is not None:

56. axes.set\_xlim(\*x\_limits)

57. if x\_ticks is not None:

58. axes.set\_xticks(x\_ticks)

59. axes.set\_xlabel(x\_label)

60. axes.set\_ylabel(y\_label)

61. axes.grid()

62. return axes

63.

64. def process(

65. implicit\_euler: ImplicitEulerType, n: int, solution\_plot: Axes,

66. ) -> FloatArray2D:

67. ts, ys, errors, dt = implicit\_euler(n)

68. data: dict[str, FloatArray] = {'t': ts, 'y': ys, 'error': errors}

69. data\_frame: DataFrame = DataFrame(data).rename\_axis('N', axis=0)

70. max\_error = errors.max()

71. n\_description: str = f'N from 0 to {n}'

72. print(f'{n\_description}\nMax error: {max\_error:.6e}\n')

73. print(f'{data\_frame.T}\n\n' + '-' \* 80 + '\n')

74. solution\_plot.plot(ts, ys, label = n\_description)

75. return array([n, log10(dt), log10(max\_error)])

76.

77. def calculate\_slope(xs: FloatArray, ys: FloatArray) -> float:

78. return (ys[-1] - ys[0]) / (xs[-1] - xs[0])

79.

80. if \_\_name\_\_ == '\_\_main\_\_':

81. # Data

82. a: float = 1.05

83. N: int = 28

84.

85. # Constants

86. t\_min: float = 0.0

87. t\_max: float = 2.0

88. y0: float = 0.0

89.

90. t\_gaps\_count: int = 10

91.

92. # Fuctions and plot preparation

93. t\_tick\_size: float = (t\_max - t\_min) / t\_gaps\_count

94. t\_ticks: FloatArray = arange(t\_min, t\_max + t\_tick\_size, t\_tick\_size)

95. solution\_plot: Axes = get\_axes(

96. 'y(t) graph', [t\_min, t\_max], t\_ticks, 't', 'y(t)',

97. )

98.

99. exact\_solution: ExactSolutionType = exact\_solution\_getter(a)

100. implicit\_euler: ImplicitEulerType = implicit\_euler\_getter(

101. a, t\_min, t\_max, y0, exact\_solution,

102. )

103.

104. # Executing implicit Euler

105. error\_plot\_data: list[FloatArray] = []

106. for n in [N, 2\*N, 4\*N]:

107. error\_plot\_data.append(process(implicit\_euler, n, solution\_plot))

108.

109. # Drawing solution

110. solution\_plot.legend()

111. if isinstance(fig := solution\_plot.get\_figure(), Figure):

112. fig.show()

113.

114. # Calculating slope

115. \_, log10\_dts, log10\_max\_errors = array(error\_plot\_data).T

116. error\_plot\_slope: float = calculate\_slope(log10\_dts, log10\_max\_errors)

117. error\_plot\_slope\_text: str = f'Slope = {error\_plot\_slope:.6f}'

118.

119. # Drawing log\_10(dt) to log\_10(max\_error) plot

120. errors\_plot: Axes = get\_axes(

121. 'log\_10(dt) to log\_10(max\_error)', None, None,

122. 'log\_10(dt)', 'log\_10(max\_error)',

123. )

124. errors\_plot.plot(

125. log10\_dts, log10\_max\_errors, marker = 'o',

126. label = error\_plot\_slope\_text,

127. )

128. for n, log10\_dt, log10\_max\_error in error\_plot\_data:

129. errors\_plot.annotate(f'N = {int(n)}', (log10\_dt, log10\_max\_error))

130.

131. errors\_plot.legend()

132. if isinstance(fig := errors\_plot.get\_figure(), Figure):

133. fig.show()

134.

135. reset\_option('display.float\_format')

136. reset\_option('display.max\_columns')

## ex\_3.py

1. from matplotlib.pyplot import figure, Axes, Figure

2. from numpy import (square, sqrt, exp, float64, arange, abs, log10, array, zeros,

3. repeat, triu, eye)

4. from numpy.typing import NDArray

5. from pandas import DataFrame, set\_option, reset\_option

6. from typing import TypeAlias, Callable, Annotated, Literal

7.

8. FloatArray: TypeAlias = Annotated[NDArray[float64], Literal[1]]

9. FloatArray2D: TypeAlias = Annotated[NDArray[float64], Literal[2]]

10. ExactSolutionType: TypeAlias = Callable[[float], float]

11. Solve2DegreeDifferentialEquationReturnType: TypeAlias = tuple[FloatArray, FloatArray, FloatArray, float]

12. Solve2DegreeDifferentialEquationType: TypeAlias = Callable[[int], Solve2DegreeDifferentialEquationReturnType]

13.

14. set\_option('display.float\_format', lambda x: '%.6e' % x)

15. set\_option('display.max\_columns', 1000)

16.

17. def exact\_solution\_getter(q: float, r: float, s: float) -> ExactSolutionType:

18. def exact\_solution(x: float) -> float:

19. negative\_half\_q: float = -q / 2

20. square\_root: float = sqrt(square(q) - 4 \* r)

21. lambda\_1: float = negative\_half\_q - square\_root / 2

22. lambda\_2: float = negative\_half\_q + square\_root / 2

23. A: float = (exp(5 \* lambda\_2) - 1) / (exp(5 \* lambda\_1) - exp(5 \* lambda\_2))

24. B: float = (exp(5 \* lambda\_1) - 1) / (exp(5 \* lambda\_2) - exp(5 \* lambda\_1))

25. return s / r \* (A \* exp(lambda\_1 \* x) + B \* exp(lambda\_2 \* x) + 1)

26. return exact\_solution

27.

28. def lower\_upper\_decomposition(

29. A: FloatArray2D,

30. ) -> tuple[FloatArray2D, FloatArray2D]:

31. n: int = A.shape[0]

32. a: FloatArray2D = A.copy()

33. for k in range(n - 1):

34. akk: float = a[k][k]

35. for i in range(k + 1, n):

36. aux: float = a[i][k] / akk if akk else 0

37. for j in range(k + 1, n):

38. a[i][j] -= a[k][j] \* aux

39. a[i][k] = aux

40. U: FloatArray2D = triu(a)

41. L: FloatArray2D = a - U + eye(n)

42. return L, U

43.

44. def eliminate\_forward(L: FloatArray2D, B: FloatArray) -> FloatArray:

45. n: int = L.shape[0]

46. b: FloatArray = B.copy()

47. for k in range(n - 1):

48. for i in range(k + 1, n):

49. b[i] -= b[k] \* L[i][k]

50. return b

51.

52. def substitute\_backward(U: FloatArray2D, Y: FloatArray) -> FloatArray:

53. n: int = U.shape[0]

54. y: FloatArray = Y.copy()

55. y[n-1] /= U[n-1][n-1]

56. for i in range(n-2, -1, -1):

57. s: float = 0.0

58. for j in range(i+1, n):

59. s += U[i][j] \* y[j]

60. y[i] -= s

61. y[i] /= U[i][i]

62. return y

63.

64. def solve\_2\_degree\_differential\_equation\_getter(

65. q: float, r: float, s: float, x\_min: float, x\_max: float,

66. y\_for\_x\_min: float, y\_for\_x\_max: float, exact\_solution: ExactSolutionType,

67. ) -> Solve2DegreeDifferentialEquationType:

68. def solve\_2\_degree\_differential\_equation(

69. N: int,

70. ) -> Solve2DegreeDifferentialEquationReturnType:

71. h: float = (x\_max - x\_min) / N

72.

73. A: FloatArray2D = zeros((N + 1, N + 1))

74. A[0][0] = 1

75. A[N][N] = 1

76.

77. B: FloatArray = repeat(s, N + 1)

78. B[0] = y\_for\_x\_min

79. B[N] = y\_for\_x\_max

80.

81. for i in arange(1, N):

82. A[i][i - 1] = 1 / square(h) - q / 2 / h

83. A[i][i] = r - 2 / square(h)

84. A[i][i + 1] = 1 / square(h) + q / 2 / h

85.

86. L, U = lower\_upper\_decomposition(A)

87. eliminated: FloatArray = eliminate\_forward(L, B)

88. ys: FloatArray = substitute\_backward(U, eliminated)

89.

90. xs: list[float] = []

91. errors = []

92. for i in arange(0, N + 1):

93. xs.append(x\_min + h \* i)

94. errors.append(abs(exact\_solution(xs[i]) - ys[i]))

95.

96. return array(xs), ys, array(errors), h

97. return solve\_2\_degree\_differential\_equation

98.

99. def get\_axes(

100. title: str, x\_limits: list[float] | None,

101. x\_ticks: FloatArray | None, x\_label: str, y\_label: str,

102. ) -> Axes:

103. axes: Axes = figure().add\_subplot()

104. axes.set\_title(title)

105. if x\_limits is not None:

106. axes.set\_xlim(\*x\_limits)

107. if x\_ticks is not None:

108. axes.set\_xticks(x\_ticks)

109. axes.set\_xlabel(x\_label)

110. axes.set\_ylabel(y\_label)

111. axes.grid()

112. return axes

113.

114. def process(

115. solve\_2\_degree\_differential\_equation: Solve2DegreeDifferentialEquationType,

116. n: int, solution\_plot: Axes,

117. ) -> FloatArray2D:

118. xs, ys, errors, h = solve\_2\_degree\_differential\_equation(n)

119. data: dict[str, FloatArray] = {'x': xs, 'y': ys, 'error': errors}

120. data\_frame: DataFrame = DataFrame(data).rename\_axis('N', axis=0)

121. max\_error = errors.max()

122. n\_description: str = f'N from 0 to {n}'

123. print(f'{n\_description}\nMax error: {max\_error:.6e}\n')

124. print(f'{data\_frame.T}\n\n' + '-' \* 80 + '\n')

125. solution\_plot.plot(xs, ys, label = n\_description)

126. return array([n, log10(h), log10(max\_error)])

127.

128. def calculate\_slope(xs: FloatArray, ys: FloatArray) -> float:

129. return (ys[-1] - ys[0]) / (xs[-1] - xs[0])

130.

131. if \_\_name\_\_ == '\_\_main\_\_':

132. # Data

133. q: float = 0.47

134. r: float = -0.84

135. s: float = 2.37

136. N: int = 28

137.

138. # Constants

139. x\_min: float = 0.0

140. x\_max: float = 5.0

141. y\_for\_x\_min: float = 0.0

142. y\_for\_x\_max: float = 0.0

143.

144. x\_gaps\_count: int = 10

145.

146. # Fuctions and plot preparation

147. x\_range: float = x\_max - x\_min

148. x\_tick\_size: float = x\_range / x\_gaps\_count

149. x\_ticks: FloatArray = arange(x\_min, x\_max + x\_tick\_size, x\_tick\_size)

150. solution\_plot: Axes = get\_axes(

151. 'y(x) graph', [x\_min, x\_max], x\_ticks, 'x', 'y(x)',

152. )

153.

154. exact\_solution: ExactSolutionType = exact\_solution\_getter(q, r, s)

155. solve\_2\_degree\_differential\_equation: Solve2DegreeDifferentialEquationType = (

156. solve\_2\_degree\_differential\_equation\_getter(

157. q, r, s, x\_min, x\_max, y\_for\_x\_min, y\_for\_x\_max, exact\_solution,

158. )

159. )

160.

161. # Executing solve 2nd degree differential equation

162. error\_plot\_data: list[FloatArray] = []

163. for n in [N, 2\*N, 4\*N]:

164. error\_plot\_data.append(

165. process(solve\_2\_degree\_differential\_equation, n, solution\_plot)

166. )

167.

168. # Drawing solution

169. solution\_plot.legend()

170. if isinstance(fig := solution\_plot.get\_figure(), Figure):

171. fig.show()

172.

173. # Calculating slope

174. \_, log10\_hs, log10\_max\_errors = array(error\_plot\_data).T

175. error\_plot\_slope: float = calculate\_slope(log10\_hs, log10\_max\_errors)

176. error\_plot\_slope\_text: str = f'Slope = {error\_plot\_slope:.6f}'

177.

178. # Drawing log\_10(dt) to log\_10(max\_error) plot

179. errors\_plot: Axes = get\_axes(

180. 'log\_10(h) to log\_10(max\_error)', None, None,

181. 'log\_10(h)', 'log\_10(max\_error)',

182. )

183. errors\_plot.plot(

184. log10\_hs, log10\_max\_errors, marker = 'o',

185. label = error\_plot\_slope\_text,

186. )

187. for n, log10\_h, log10\_max\_error in error\_plot\_data:

188. errors\_plot.annotate(f'N = {int(n)}', (log10\_h, log10\_max\_error))

189.

190. errors\_plot.legend()

191. if isinstance(fig := errors\_plot.get\_figure(), Figure):

192. fig.show()

193.

194. reset\_option('display.float\_format')

195. reset\_option('display.max\_columns')