# Projekt – Narzędzia programistyczne


### Autorzy:

Mateusz Birkholz

Filip Sawiński

Ahmed Chazbijewicz


### Nazwa projektu:

Metoda jawna Eulera, niejawna Eulera oraz metoda trapezów

### Rok akademicki i grupa:

2019/2020 semestr letni, grupa 1.

## Informacje ogólne – użyte technologie, sposób użytkowania itp.

W celu stworzenia programów zostałe użyte:
- Python 3
- Tkinter
- NumPy

W celu stworzenia algorytmów wykorzystaliśmy stronę draw.io

Sposób wprowadzania danych:
- Dane liczbowe uzupełniamy w normalny sposób (tj. wprowadzamy liczby), natomiast liczby zmiennoprzecinkowe rozdzielamy kropką.
- Wszystkie podstawowe działania wpisujemy normalnie, natomiast potęgowanie wpisujemy za pomocą podwójnej gwiazdki (czyli np. x**3 to $x^3$), a pierwiastek kwadratowy wprowadzamy za pomocą sqrt(x)
- Bardzo istotne jest prawidłowe rozmieszczenie nawiasów. Program może mieć problem z kolejnościa ich wykonywania (czyli np. x*y+2x*3y wpiszemy jako (x*y)+(2x*3y))
- Program może przyjmować funkcje trygonometryczne (np. sin(x)).

Po uruchomieniu programu wszystkie programy zostały uzupełnione przykładowymi danymi która można znaleźć również w dalszej części dokumentacji w celu pokazania faktu, że nasze programy działają poprawnie w stosunku do rozwiązywania problemów ręcznie. Dane te można oczywiscie zmieniać wedle uznania.

Cały projekt zawiera 3 programy: Rozwiązujący problemy jawną metodą Eulera, niejawną metodą Eulera oraz metodą trapezów. W celu poznania co jaka dana oznacza, należy przejść do odpowiedniej sekcji w dokumentacji.

## Jawna metoda Eulera – rozwiązanie przykładowego zadania

Dzięki jawnej metodzie Eulera możemy obliczyć jaki jest y w danym miejsu funkcji dla podanych wzorów, x początkowego oraz y poczatkowego. Warto dodać, że zarówno jawna i niejawna jest raczej niedokładna (w zależności od wielkości kroku o którym zaraz powiem) i działa tylko na równania z dwoma niewiadomymi. Załóżmy że mamy równanie:

*f(x, y) = x + 2y*

i chcemy policzyć jaki jest y, kiedy x jest równy 0.4. Na początku musimy sobie wybrać h. Czym jest h? Wielkością kroku, który wybieramy sobie sami. Im h mniejsze, tym wynik będzie dokładniejszy. Przy wyborze h musimy pamiętać, żeby x było podzielne przez h, czyli np. możemy sobie wziąć h = 0.1 bo dzieli x=0.4, ale h=0.09 już nie. Musimy sobie również wybrać od jakiego x chcemy zacząć (krok początkowy). Zazwyczaj zaczyna się od 0, a więc zakładam że $x_0 = 0$. A więc mamy dane:

f(x, y) = x + 2y
x=0.4
h=0.1
$x_0 = 0$
y=? (szukana)

Aby znaleźć y, najpierw musimy zająć się x-ami i krokami. Potrzebny jest nam wzór:

$$x_n = x_{n-1} + h$$

Używamy tego wzoru dopóki nie dojdziemy do x=0.4. A więc:

| $x_n$ | Obl + wynik |
|---|---|
| $x_0$ | **0** |
| $x_1$ | 0+0.1=**0.1** |
| $x_2$ | 0.1+0.1=**0.2** |
| $x_3$ | 0.2+0.1=**0.3** |
| $x_4$ | 0.3+0.1=**0.4** |

Gdy już obliczyliśmy wszystkie potrzebne x, czas przejść do policzenia y odpowiadających danym x. Wynikiem całego działania będzie w tym przypadku $y_4$. Aby policzyć ten y, Musimy użyć wzoru

$$y_n = y_{n-1} + h * f(x_{n-1}, y_{n-1})$$

Robimy to samo co poprzednio, tylko że dla y. Jakie jest $y_0$? Tak samo jak $x_0$, wybieramy sobie, i tak samo jak w poprzednim przypadku, najlepiej wybrać 0. A więc:

| $x_n$ | Obl + wynik |
|---|---|
| $x_0$ | **0** |
| $x_1$ | 0+0.1=**0.1** |
| $x_2$ | 0.1+0.1=**0.2** |
| $x_3$ | 0.2+0.1=**0.3** |
| $x_4$ | 0.3+0.1=**0.4** |

*f(x, y) = x + 2y*

$$y_n = y_{n-1} + h * f(x_{n-1}, y_{n-1})$$

| $y_n$ | Obl + wynik |
|---|---|
| $y_0$ | **0** |
| $y_1$ | 0+0.1*(0+0)=**0** |
| $y_2$ | 0+0.1*(0.1+0)=**0.01** |
| $y_3$ | 0.01+0.1*(0.2+0.02)=**0.032** |
| $y_4$ | 0.032+0.1*(0.3+0.064)=**0.0684** |

Po policzeniu ostatniego y, mamy wynik: **0.0684**, co jest rozwiązaniem naszego zadania.

*<u>W programie użytkownik podaje: Równanie, x0, y0, x docelowe (x=0.4 w tym przypadku) oraz h.</u>*

## Jawna metoda Eulera – działanie programu

Program, tak jak napisano we wcześniejszym rozdziale:
<u>- Przyjmuje od użytkownika:</u>
Równanie
x0, y0
x docelowe
Wielkość kroku (h)
<u>- Zwraca:</u>
y docelowy w zależności od docelowego x

**Krok 1: Weryfikacja prawidłowości podanych danych:** Na początku program sprawdza czy otrzymał wszystkie informacje. Wtedy program sprawdza, czy (x docelowy + x0) jest podzielne przez krok.

**Krok 2:** Po zweryfikowaniu prawidłowości danych, program tworzy tabelę x-ów, w którą najpierw wstawia początkowego x, a potem uruchamia pętle dzięki której uzupełnia resztę x-ów w taki sam sposób, w jaki zostało to zrealizowane w podanym zadaniu wyżej.

**Krok 3:** Gdy program policzy wszystkie x, przechodzi do policzenia kolejnych y. Najpierw program wpisuje w nowo stworzoną tabelę y0, po czym liczy kolejne y tak samo jak w zadaniu podanym powyżej. Jest to możliwe dzięki stworzeniu w poprzednim kroku tabelę która zawiera kolejne x.

**Krok 4:** Program zwraca ostatni y z tej tabeli, a więc wynik.

## Niejawna metoda Eulera – rozwiązanie przykładowego zadania

Główny wzór w niejawnej metodzie Eulera to:

$$y_n = y_{n-1} + h * f(x_n, y_n)$$

Ta metoda jest nazywana również wsteczną metodą Eulera. W niejawnej metodzie Eulera otrzymujemy trochę inny typ problemu. Jak w jawnej metodzie Eulera dane było polecenie typu „jaki jest y kiedy jakiś y(-) jest równy...” to tutaj otrzymujemy polecenie typu
„Kiedy $y_1(0) = \cdots$ & $y_2(0) = \cdots$...” (przynajmniej w przypadku naszego programu, który rozwiązuje ten typ zadania). Tutaj będziemy potrzebowali rozwiązać układ równań. Załóżmy że mamy dwa równania:

$$f(x, y_1) = xy_1 - 3y_2$$
$$g(x, y_2) = 3y_1 - x^2 y_2$$

I polecenie: dla $y_1(0) = 2$ & $y_2(0) = 1$ & $h = 1$, oblicz $y_1\ i\ y_2$ dla x=1.
Po podstawieniu do $y_n = y_{n-1} + h * f(x_n, y_n)$ otrzymamy:

$$y_1(1) = 2 + 1 * \left(1 * y_1(1) - 3y_2(1)\right)$$
$$y_2(1) = 1 + 1 * (3 * y_1(1) - 1^2 y_2(1))$$

Dla lepszej czytelności zamienię $y_1(1)$ na c, a $y_2(1)$ na z. Po takiej zamianie otrzymamy:

$$c = 2 + 1 * (1 * c - 3z)$$
$$z = 1 + 1 * (3 * c - 1^2 z)$$

Można zauważyć, że z tych równań można ułożyć układ równań:

$$\begin{cases} c = 2 + c - 3z \\ z = 1 + 3c - z \end{cases}$$

Po rozwiązaniu układu równań otrzymuje $c = \frac{1}{9}, z = \frac{2}{3}$, a więc $y_1(1) = \frac{1}{9}, y_2(1) = \frac{2}{3}$, i to jest rozwiązanie naszego zadania.
*Użytkownik tutaj podaje: Oba równania, $y_1(0)$, $y_2(0), h\ i\ x$.*

## Niejawna metoda Eulera – działanie programu

Program, tak jak napisano we wcześniejszym rozdziale:
<u>- Przyjmuje od użytkownika:</u>
$y_1(0),\ y_2(0)$
h
x
Dwa równiania
<u>- Zwraca:</u>
$y_1(1), y_2(1)$

**Krok 1: Weryfikacja prawidłowości podanych danych:** Program po prostu sprawdza czy wszystkie dane zostały wpisane.

**Krok 2;** Program wprowadzone dane podstawia do wzoru, po czym oba wzory „przepisuje” do macierzy aby obliczyć układ równań w celu otrzymania y1 i y2.

**Krok 3:** Program zwaraca y1(1) i y2(1).

## Metoda trapezów – działanie programu

Program:
- Przyjmuje od użytkownika:
Wzór funkcji
Zakres (x1, x2) dla jakich pole pod wykresem ma być policzone
Ilość trapezów (n)
- Zwraca:
Pole pod wykresem

**Krok 1: Weryfikacja prawidłowości podanych danych:** Program po prostu sprawdza czy wszystkie dane zostały wpisane.

**Krok 2:** Program oblicza krok na podstawie dzielenia zakresu przez liczbę trapezów

**Krok 3:** Program tworzy zmienną która będzie przechowywać sumę pól trapezów, xstart która przechowuje x0 i xnext która przechowuje wartość xstart zwiększoną o krok

**Krok 4:** Program uruchamia pętle, która wykonuje się tak długo, jak xstart jest mniejszy od końca zakresu. W pętli program oblicza a i b na podstawie wartości funkcji w danym x (a – wartość funkcji w xstart, b – wartość funkcji w xnext). Na podstawie a, b i kroku liczy pole trapezu i dodaje je do sumy pól. Program zwiększa xstart i xnext o h i zaokrągla je do 6 miejsca do przecinku w celu uniknięcia błędów w programie spowodowanych ograniczeniami w liczbach.

**Krok 5:** Program wyświetla sumę pól.