Wydział	Imię i nazwisko	Rok	Grupa
WIMiIP	Mateusz Witkowski	II	4
Temat: Rozwiązywanie równań różniczkowych – metoda Eulera.			**Prowadzący** dr hab. inż. Hojny Marcin, prof. AGH
Data ćwiczenia 21.05.2020	**Data oddania** 27.05.2020	**Data zaliczenia**	**OCENA**

1. Cel ćwiczenia

Celem ćwiczenia było zapoznanie się oraz implementacja sposobu rozwiązywania równań różniczkowych zwyczajnych - metody Eulera.

2. Wprowadzenie teoretyczne

Metoda Eulera jest najprostszym numerycznym sposobem rozwiązywania równań różniczkowych. Opiera się ona o geometryczną interpretację równania różniczkowego, gdzie równanie to dla każdego punktu (x,y) określa nachylenie stycznej do rozwiązania, które przechodzi przez dany punkt. Natomiast kierunek stycznej zmienia się w sposób ciągły od punktu do punktu.

Posługując się definicją pochodnej jesteśmy w stanie wyznaczyć wzór na kolejne wartości przybliżające funkcję $x(t)$.

Definicja pochodnej:

$$\frac{dy}{dx} \equiv \lim_{h\to 0} \frac{y(x+h)-y(x)}{h} \approx \frac{y(x+h)-y(x)}{h}$$

Przekształcone wyrażenie:

$$y(x+\ \Delta x) = y(x) + \Delta x * f(x, y(x))$$

Gdzie:

- $f\big(x, y(x)\big) = \frac{dy}{dx}$
- $\Delta x = h$

By wyznaczyć kolejne przybliżenia szukanej funkcji, konieczne jest podanie rozwiązania w x_0 tak, że $y(x_0) = y_0$, dzięki temu korzystając z powyższego wzoru jesteśmy w stanie wyliczyć kolejne rozwiązania. Trzeba również przyjąć, że wartość $\frac{dy}{dx}$ jest stała między kolejnymi punktami. Równanie jest aproksymowane łamaną o następujących wierzchołkach: $(x_0, y_0), (x_0 + h, y_1), (x_0 + 2h, y_2), \dots, (x_0 + nh, y_n)$.

3. Kod programu

Zdefiniowano funkcję pomocniczą zwracającą wartość pochodnej w przekazanym do niej punkcie, co znacznie ułatwi nam pracę w przypadku modyfikacji kodu, gdyż będziemy chcieli rozpatrywać wiele różnych funkcji.

```
double f(double x, double y) {  //Funkcja pomocnicza
    return y;
}
```

Rysunek 1. Funkcja wartości pochodnej.

Następnie zaimplementowano algorytm Eulera w postaci funkcji przyjmującej za zmienne punkt początkowy – współrzędną x i y, kraniec interesującego nas przedziału oraz wielkość kroku wyznaczająca kolejne punkty. Na podstawie punktu początkowego, krańca przedziału i wielkości kroku wyliczono wartość N, czyli potrzebną liczbę iteracji do wykonania algorytmu. Utworzono pętlę for odpowiedzialną za wyliczanie i wypisywanie kolejnych x i y.

```
void Euler(double a, double b, double y0, double h) {
    double N = (b - a) / h; //wyliczamy liczbe iteracji
    double x = a;   //przypisujemy x punkt startowy
    double y = y0;  //przypisujemy parametr poczatkowy
    for (size_t i = 0; i < N; i++)
    {
        y = y + h * f(x, y);    //wyliczamy i wypisujemy kolejne y
        cout << "y[" << i + 1 << "] = " << y << endl;
        x = x + h;              //wyliczamy i wypisujemy kolejne x
        cout << "x[" << i + 1 << "] = " << x << endl;
        cout << endl;
    }
}
```

Rysunek 2. Implementacja metody Eulera.

W funkcji main przed i po wywołaniu funkcji „Euler” zdefiniowano zmienne pobierające moment czasowy w celu obliczenia oraz wyświetlenia czasu potrzebnego do zrealizowania algorytmu.

```
int main() {

    auto start = std::chrono::steady_clock::now();      //punkt startowy pomiaru czasu
    Euler(0,3,1,0.5);
    auto end = std::chrono::steady_clock::now();        //punkt koncowy pomiaru czasu
    std::chrono::duration<double, milli> elapsed_seconds = end - start;     //wyliczamy czas ktory uplynal
    std::cout << "Czas wykonania algorytmu: " << elapsed_seconds.count() << "ms\n";

    getchar(); getchar();
    return 0;
}
```

Rysunek 3. Funkcja main.

Cały kod:

```
double f(double x, double y) {  //Funkcja pomocnicza
    return y;
}
void Euler(double a, double b, double y0, double h) {
    double N = (b - a) / h; //wyliczamy liczbe iteracji
    double x = a;   //przypisujemy x punkt startowy
    double y = y0;  //przypisujemy parametr poczatkowy
    for (size_t i = 0; i < N; i++)
    {
        y = y + h * f(x, y);    //wyliczamy i wypisujemy kolejne y
        cout << "y[" << i + 1 << "] = " << y << endl;
        x = x + h;              //wyliczamy i wypisujemy kolejne x
        cout << "x[" << i + 1 << "] = " << x << endl;
        cout << endl;
    }
}
int main() {

    auto start = std::chrono::steady_clock::now();      //punkt startowy pomiaru czasu
    Euler(0,3,1,0.5);
    auto end = std::chrono::steady_clock::now();        //punkt koncowy pomiaru czasu
    std::chrono::duration<double, milli> elapsed_seconds = end - start;     //wyliczamy czas ktory uplynal
    std::cout << "Czas wykonania algorytmu: " << elapsed_seconds.count() << "ms\n";

    getchar(); getchar();
    return 0;
}
```

Rysunek 4. Cały kod programu.

4. Testy

W celu zweryfikowania wyników programu dokonano testów na podanych w instrukcji parametrach oraz na własnym równaniu różniczkowym. Wszystkie wyniki porównano z rozwiązaniami dokładnymi oraz wykreślono odpowiednie wykresy przy użyciu programu Microsoft Excel.

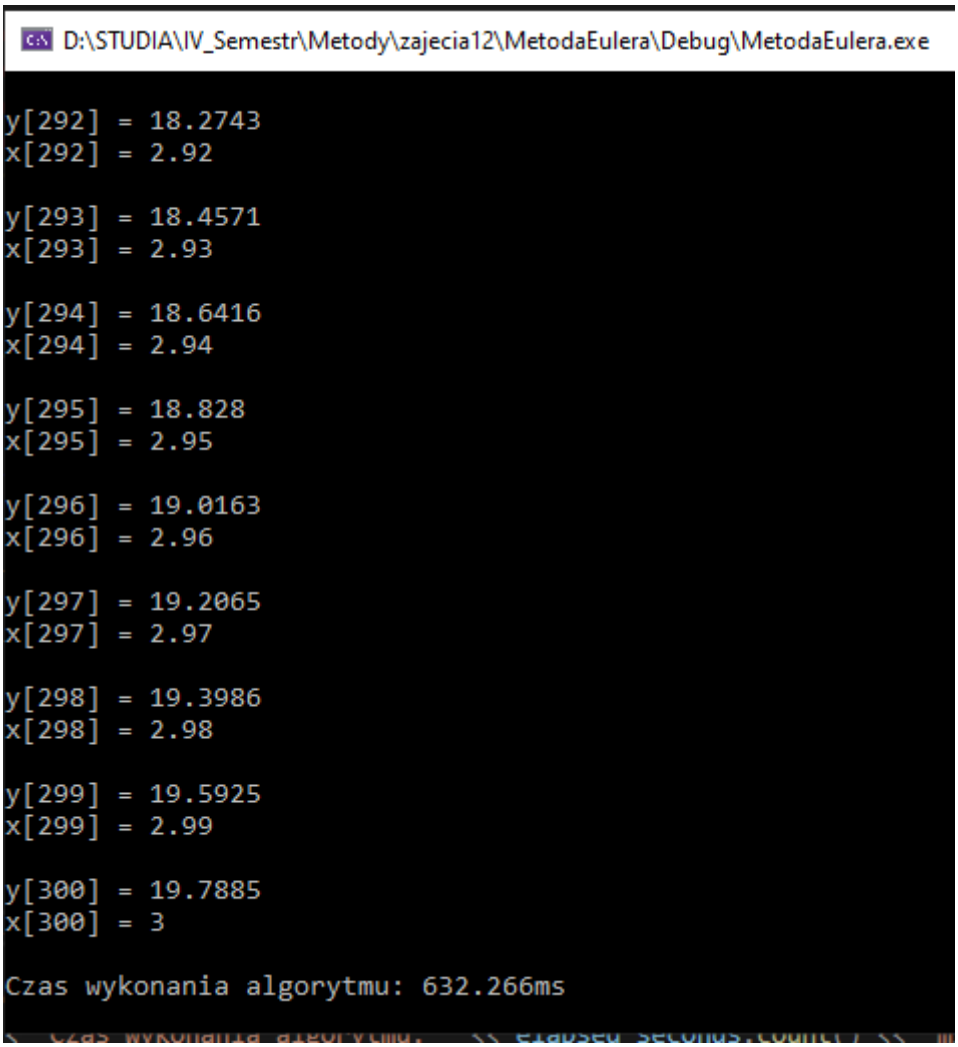
Test 1 – Równanie różniczkowe z instrukcji.

Przypadek A.

Tabela 1. Dane do test 1, przypadku A.

x_0	y_0	b	h	$\frac{dy}{dx}$	Rozwiązanie analityczne
0	1	3	0.01	y	e^x

Wartości zwrócone przez program:



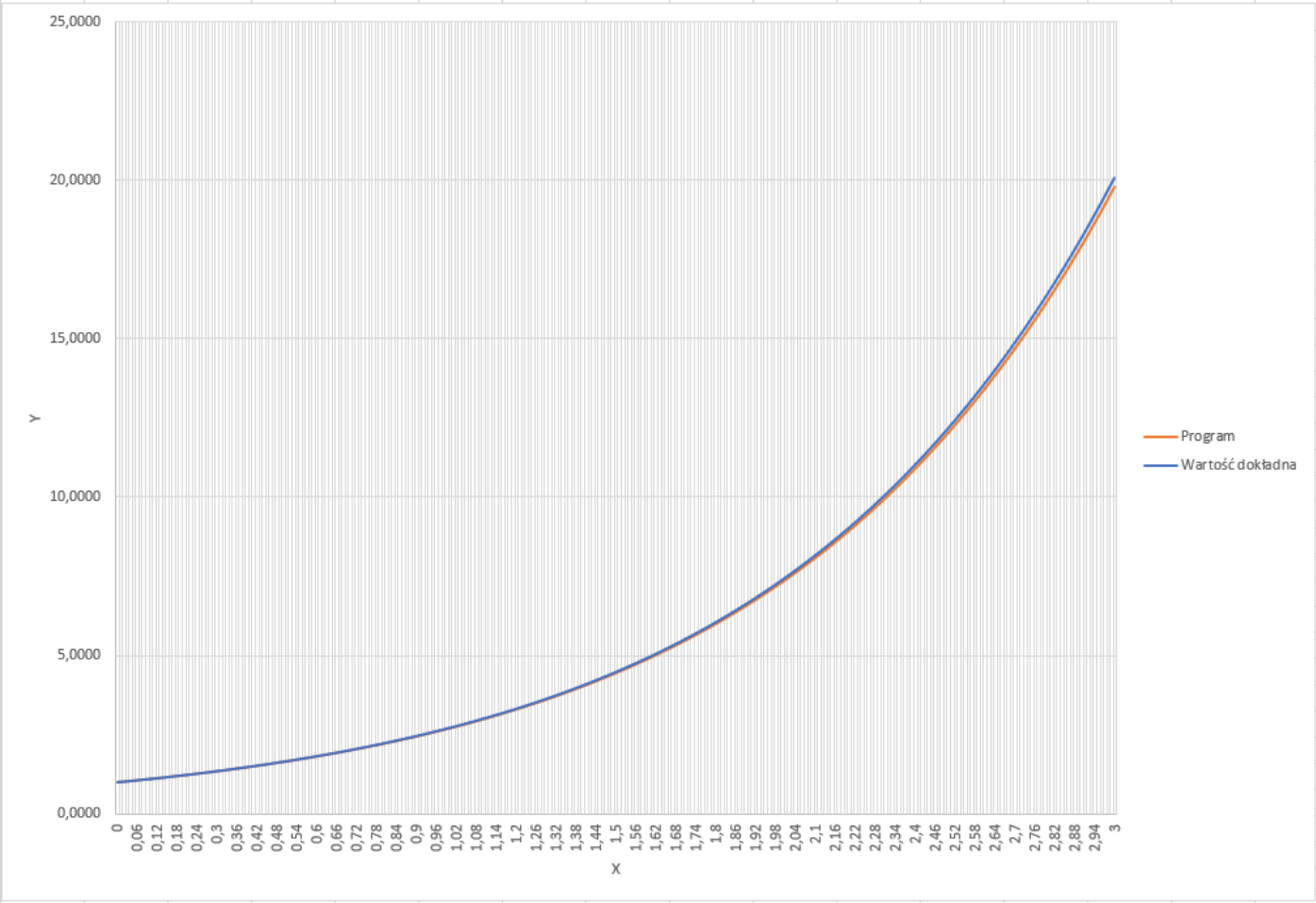
Rysunek 5.Końcowe wartości zwrócone przez program dla h = 0.01.

Wartości obliczone w Excelu:

x290	2,9		y290	17,9142		y290	18,17415
x291	2,91		y291	18,0934		y291	18,3568
x292	2,92		y292	18,2743		y292	18,54129
x293	2,93		y293	18,4571		y293	18,72763
x294	2,94		y294	18,6416		y294	18,91585
x295	2,95		y295	18,8280		y295	19,10595
x296	2,96		y296	19,0163		y296	19,29797
x297	2,97		y297	19,2065		y297	19,49192
x298	2,98		y298	19,3986		y298	19,68782
x299	2,99		y299	19,5925		y299	19,88568
x300	3		y300	19,7885		y300	20,08554
Punkty			Program			EXP(X)	

Rysunek 6. Wyniki otrzymane w Excelu.

Wykres:



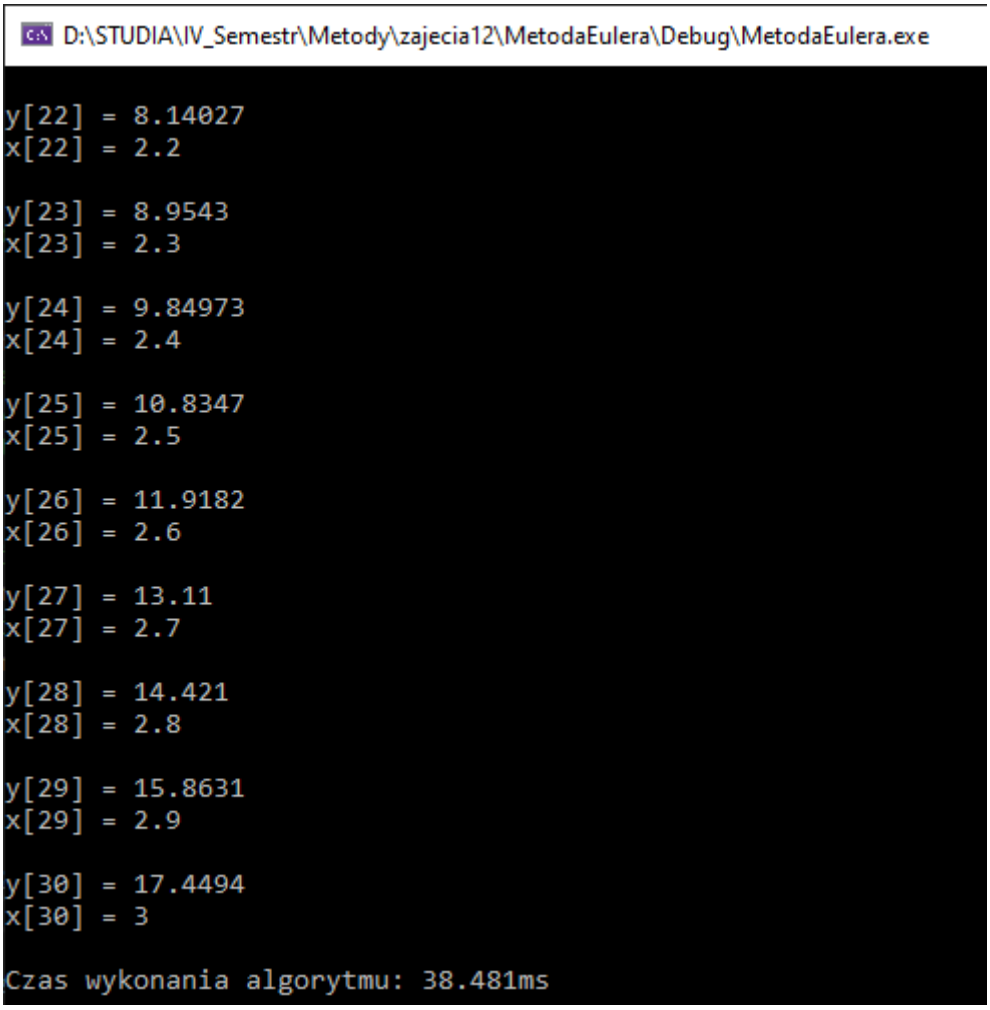
Rysunek 7. Wykres porównujący wartości wyliczone w programie z wartościami dokładnymi.

Przypadek B.

Tabela 2.Dane do test 1, przypadku B.

x_0	y_0	b	h	$\frac{dy}{dx}$	Rozwiązanie analityczne
0	1	3	0.1	y	e^x

Wartości zwrócone przez program:



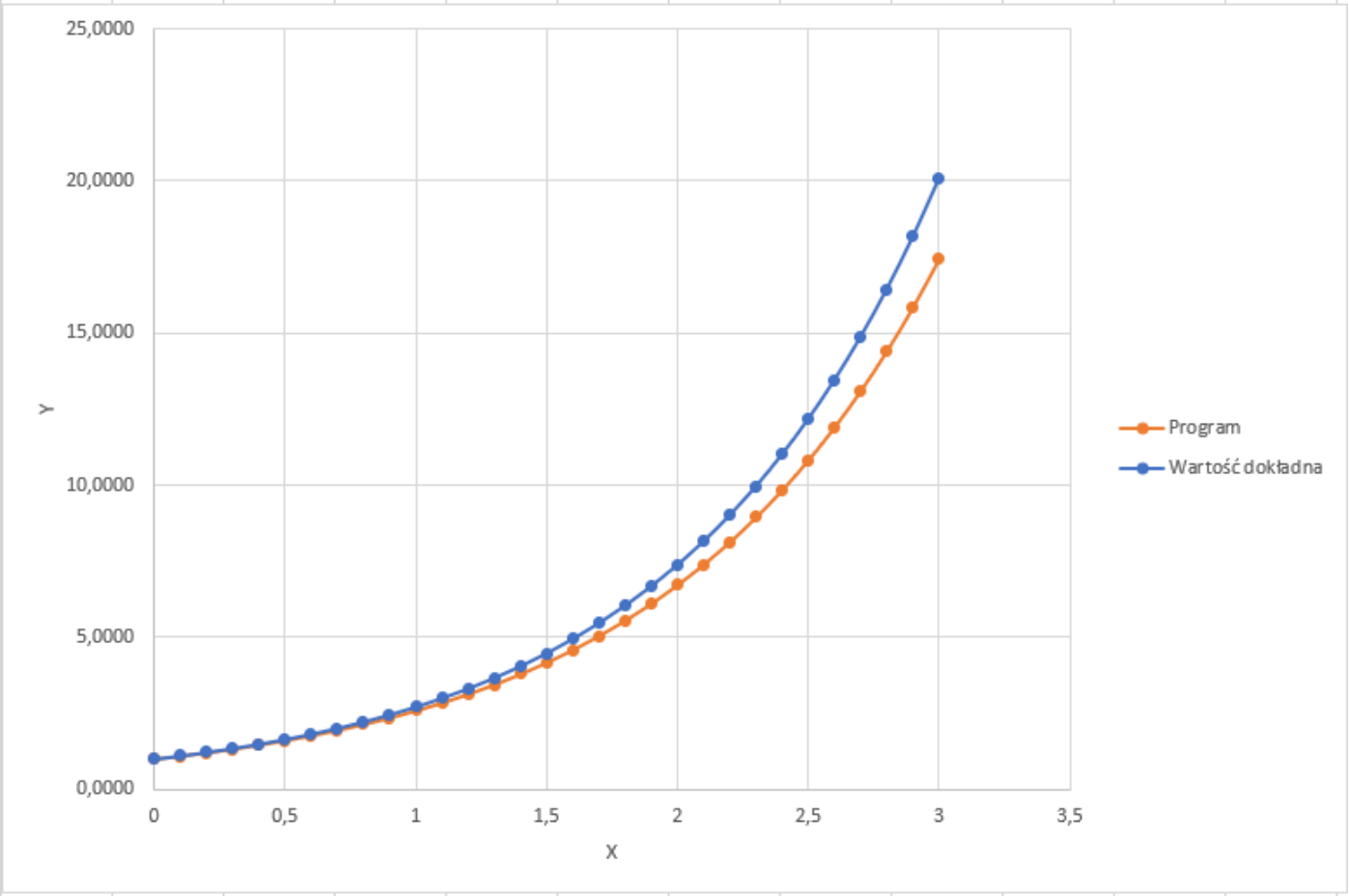
Rysunek 8.Końcowe wartości zwrócone przez program dla h = 0.1.

Wartości obliczone w Excelu:

x20	2	y20	6,7275	y20	7,389056
x21	2,1	y21	7,4002	y21	8,16617
x22	2,2	y22	8,1403	y22	9,025013
x23	2,3	y23	8,9543	y23	9,974182
x24	2,4	y24	9,8497	y24	11,02318
x25	2,5	y25	10,8347	y25	12,18249
x26	2,6	y26	11,9182	y26	13,46374
x27	2,7	y27	13,1100	y27	14,87973
x28	2,8	y28	14,4210	y28	16,44465
x29	2,9	y29	15,8631	y29	18,17415
x30	3	y30	17,4494	y30	20,08554
Punkty		Program		EXP(X)	

Rysunek 9.Wyniki otrzymane w Excelu.

Wykres:



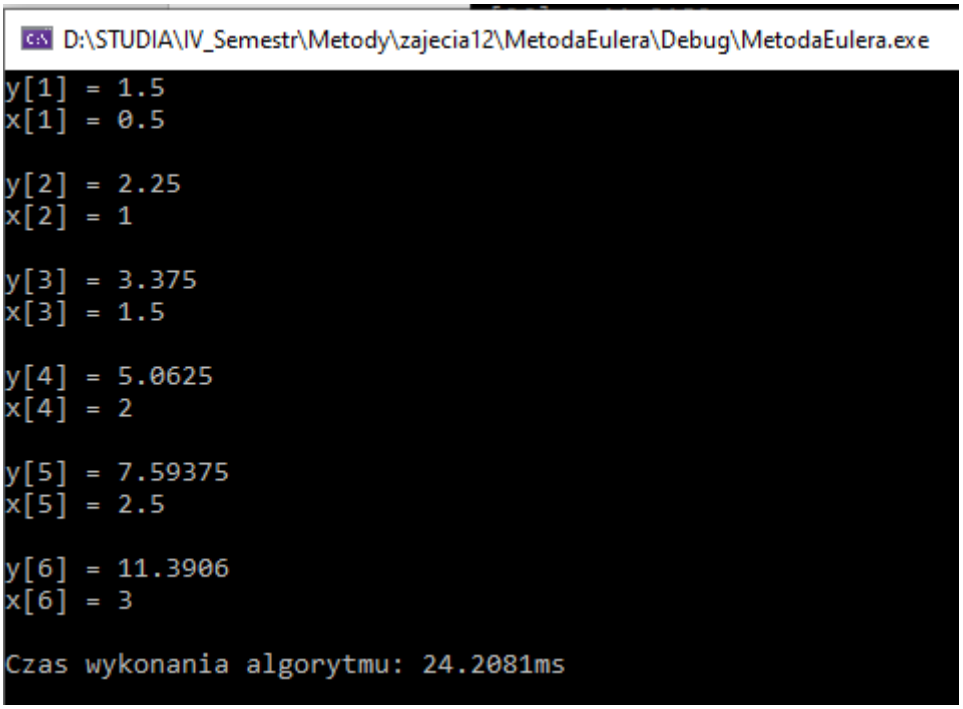
Rysunek 10.Wykres porównujący wartości wyliczone w programie z wartościami dokładnymi.

Przypadek C.

Tabela 3.Dane do test 1, przypadku C.

x_0	y_0	b	h	$\frac{dy}{dx}$	Rozwiązanie analityczne
0	1	3	0.5	y	e^x

Wartości zwrócone przez program:



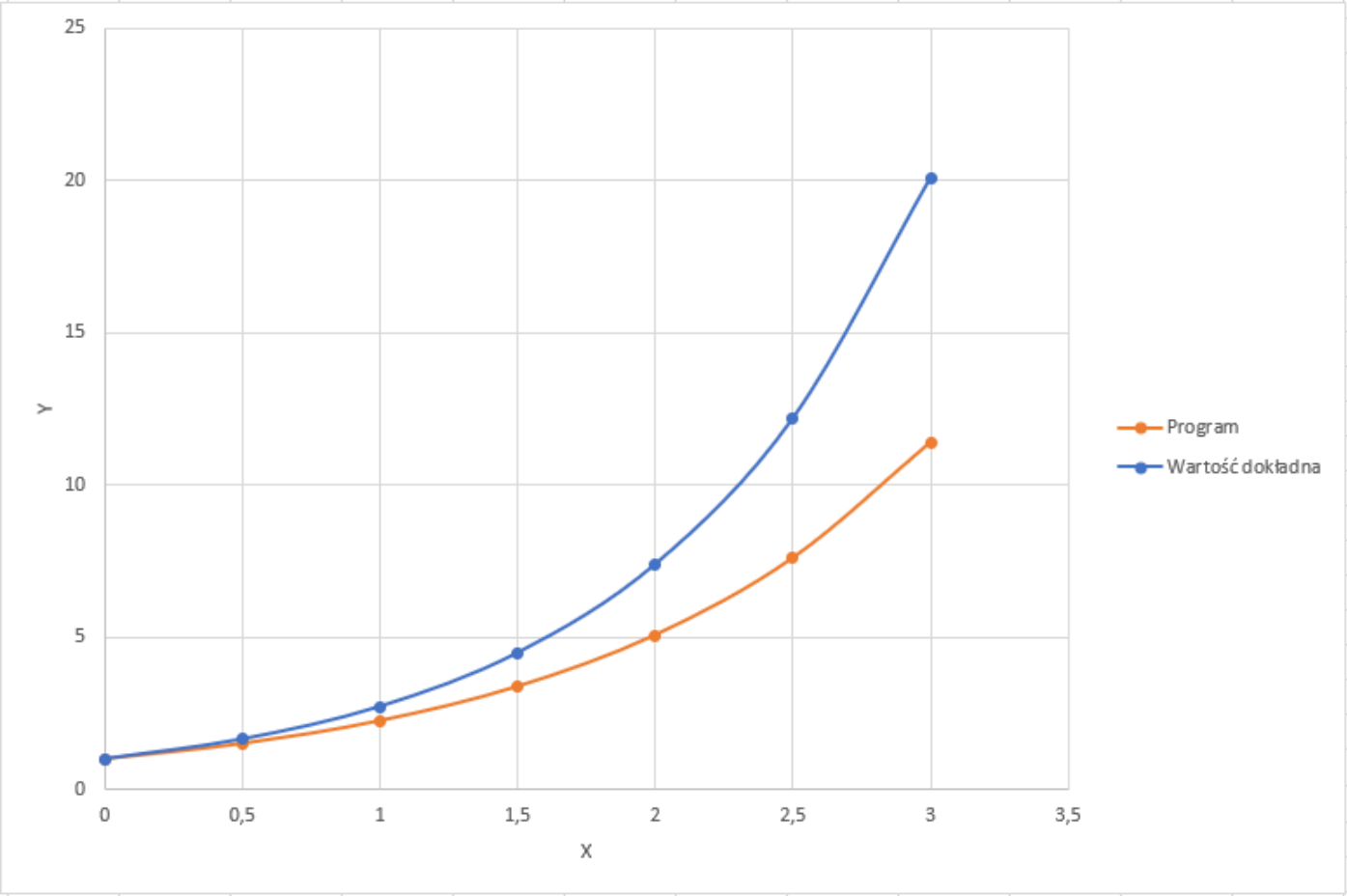
Rysunek 11.Wszystkie wartości zwrócone przez program dla h = 0.5.

Wartości obliczone w Excelu:

x0	0	y0	1	y0	1
x1	0,5	y1	1,5	y1	1,648721
x2	1	y2	2,25	y2	2,718282
x3	1,5	y3	3,375	y3	4,481689
x4	2	y4	5,0625	y4	7,389056
x5	2,5	y5	7,59375	y5	12,18249
x6	3	y6	11,39063	y6	20,08554
Punkty		Program		EXP(X)	

Rysunek 12. Wyniki otrzymane w Excelu.

Wykres:



Rysunek 13.Wykres porównujący wartości wyliczone w programie z wartościami dokładnymi.

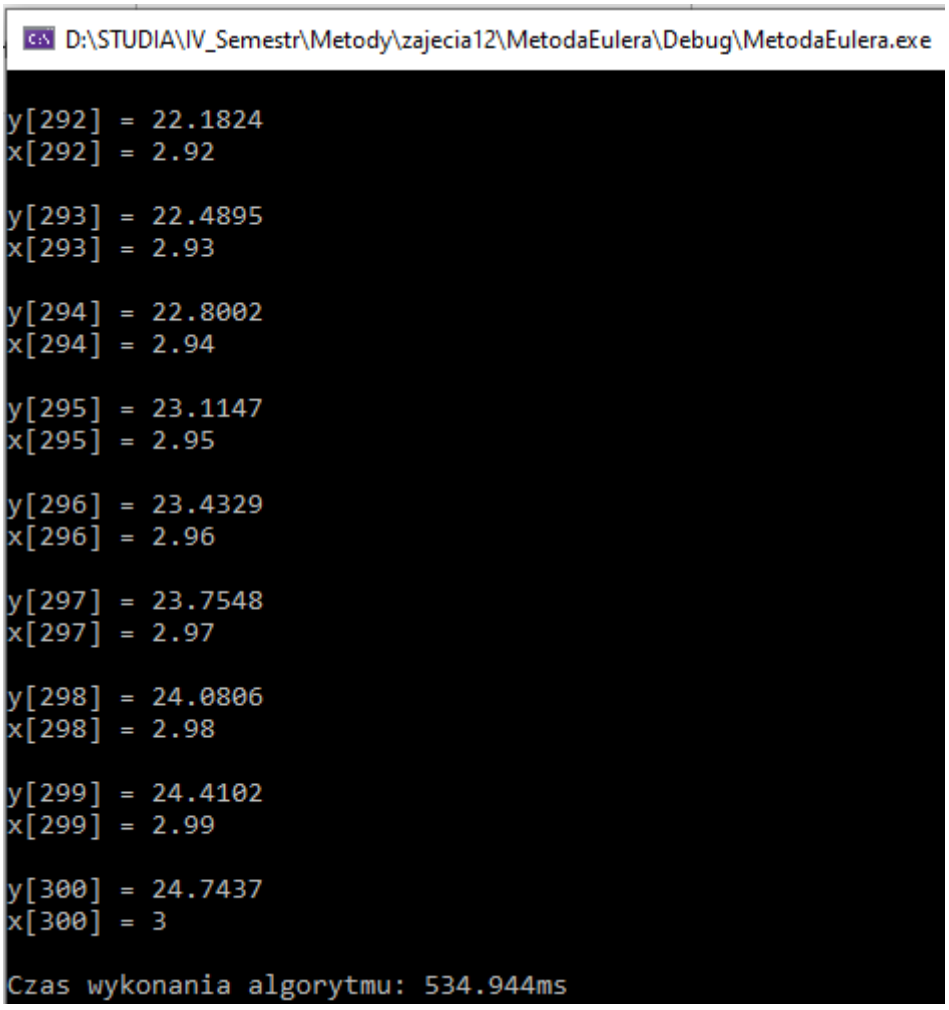
Test 2 – Własne równanie różniczkowe.

Przypadek A.

Tabela 4.Dane do test 2, przypadku A.

x_0	y_0	b	h	$\frac{dy}{dx}$	Rozwiązanie analityczne
0	0.1	3	0.01	$x^2 + y$	$2{,}1 * e^x - x^2 - 2x - 2$

Wartości zwrócone przez program:



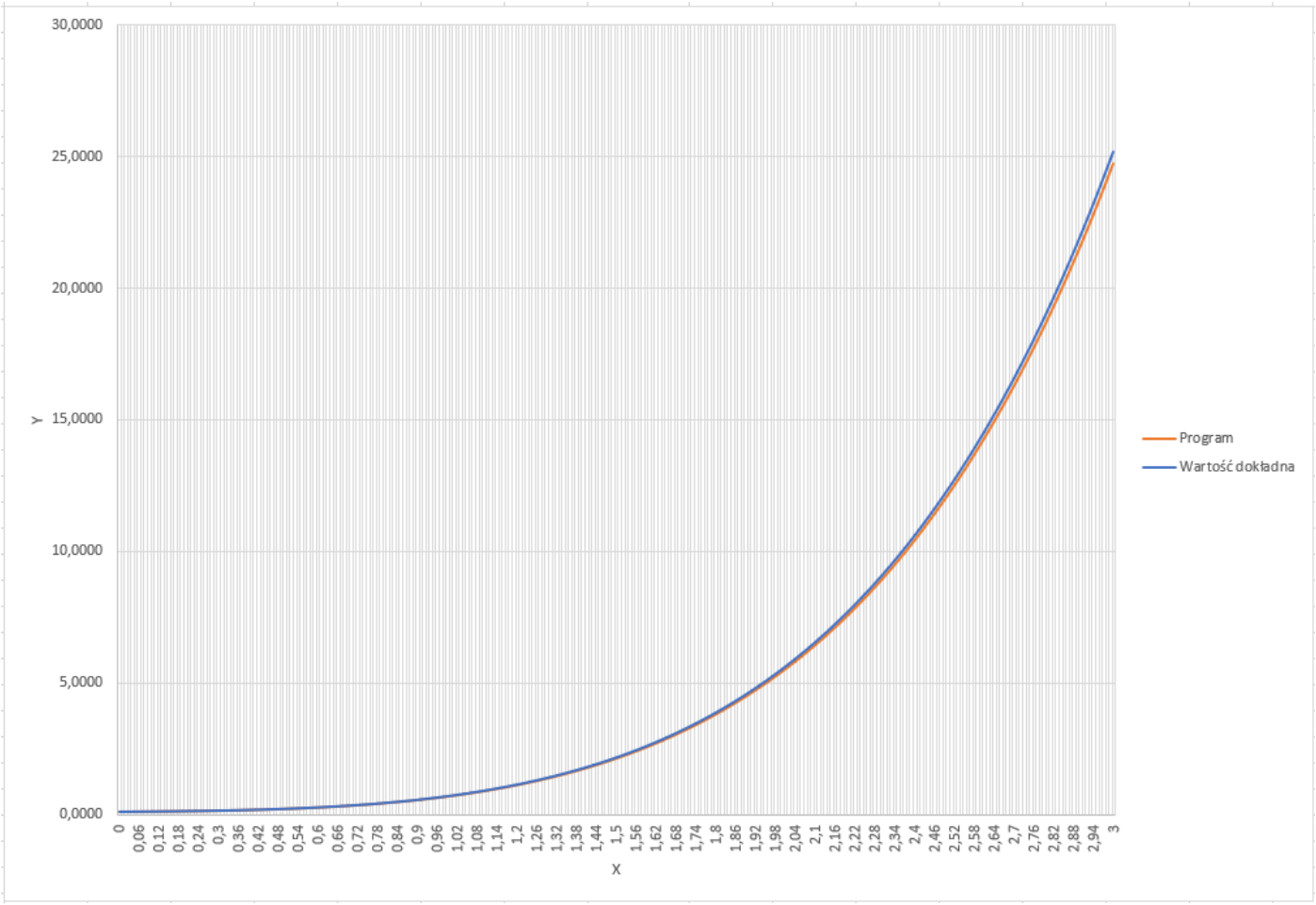
Rysunek 14.Końcowe wartości zwrócone przez program dla h = 0.01.

Wartości obliczone w Excelu:

x290	2,9	y290	21,5790	y290	21,9557053
x291	2,91	y291	21,8789	y291	22,261177
x292	2,92	y292	22,1824	y292	22,5703037
x293	2,93	y293	22,4895	y293	22,883124
x294	2,94	y294	22,8002	y294	23,1996773
x295	2,95	y295	23,1147	y295	23,5200028
x296	2,96	y296	23,4329	y296	23,8441407
x297	2,97	y297	23,7548	y297	24,1721312
x298	2,98	y298	24,0806	y298	24,504015
x299	2,99	y299	24,4102	y299	24,8398332
x300	3	y300	24,7437	y300	25,1796275
Punkty		Program		2,1*EXP(X) - x^2 -2x -2	

Rysunek 15. Wyniki otrzymane w Excelu.

Wykres:



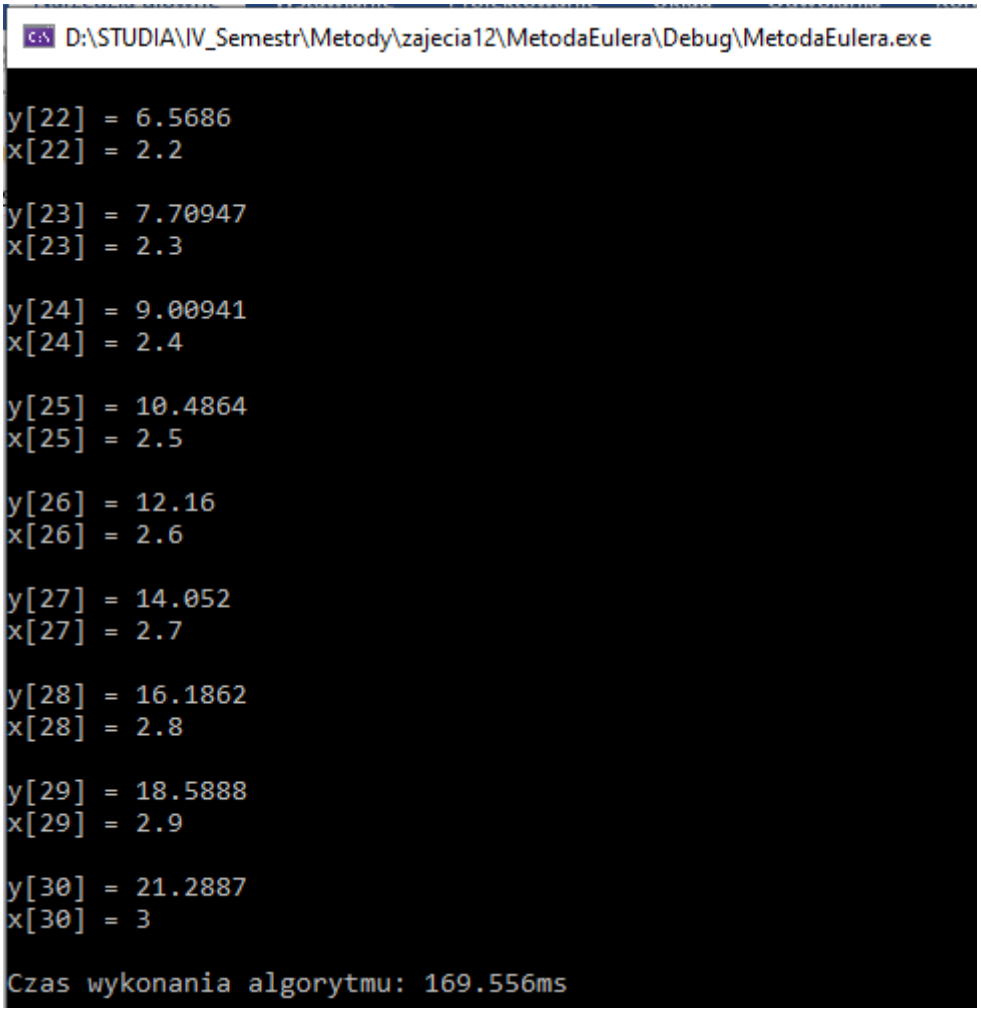
Rysunek 16.Wykres porównujący wartości wyliczone w programie z wartościami dokładnymi.

Przypadek B.

Tabela 5.Dane do test 2, przypadku B.

x_0	y_0	b	h	$\frac{dy}{dx}$	Rozwiązanie analityczne
0	0.1	3	0.1	$x^2 + y$	$2{,}1 * e^x - x^2 - 2x - 2$

Wartości zwrócone przez program:



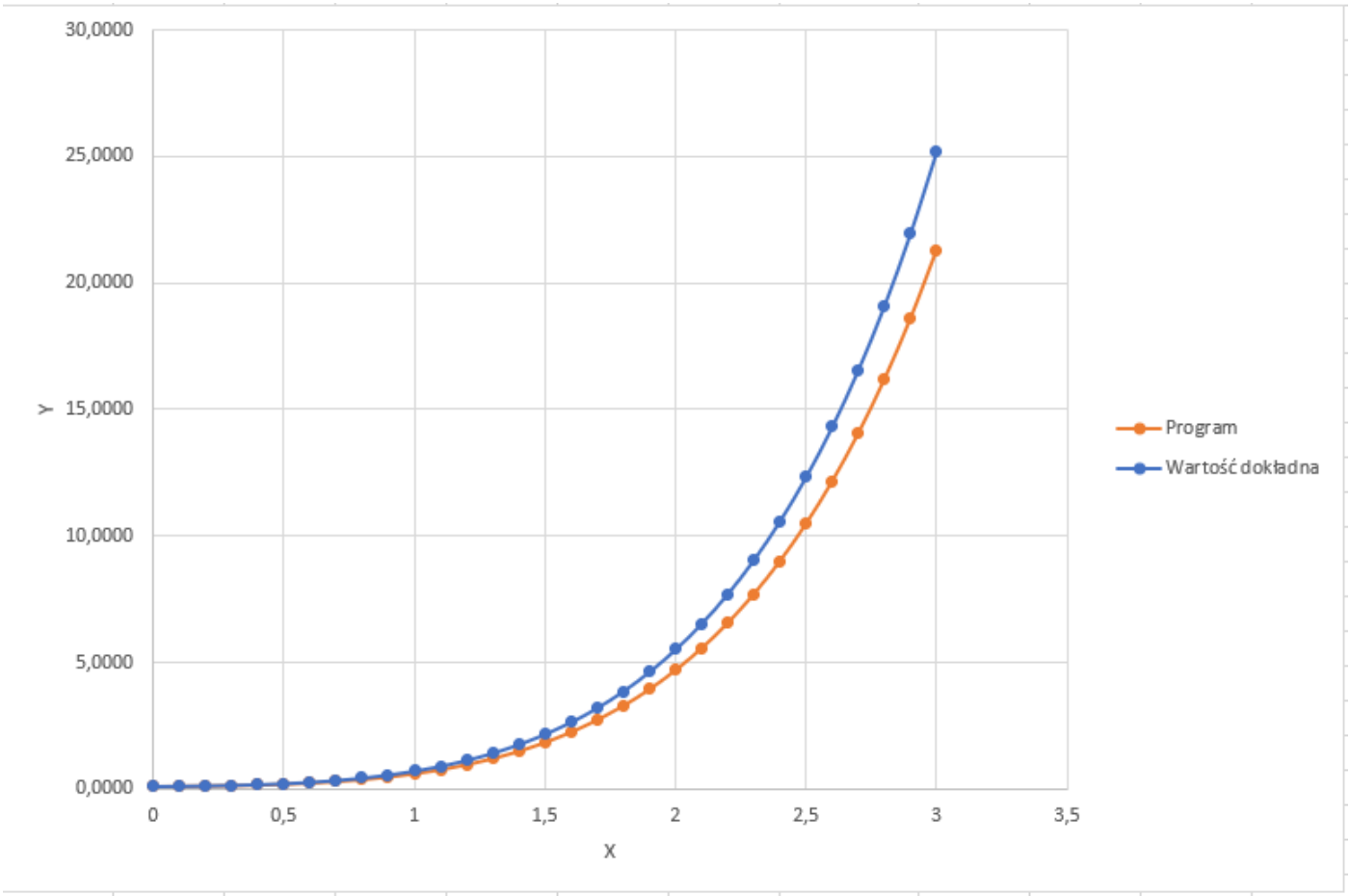
Rysunek 17.Końcowe wartości zwrócone przez program dla h = 0.1.

Wartości obliczone w Excelu:

x20	2	y20	4,7005	y20	5,51701781
x21	2,1	y21	5,5705	y21	6,53895682
x22	2,2	y22	6,5686	y22	7,71252835
x23	2,3	y23	7,7095	y23	9,05578316
x24	2,4	y24	9,0094	y24	10,5886704
x25	2,5	y25	10,4864	y25	12,3332373
x26	2,6	y26	12,1600	y26	14,3138499
x27	2,7	y27	14,0520	y27	16,5574366
x28	2,8	y28	16,1862	y28	19,0937582
x29	2,9	y29	18,5888	y29	21,9557053
x30	3	y30	21,2887	y30	25,1796275
Punkty		Program		2,1*EXP(X) - x^2 -2x -2	

Rysunek 18.Wyniki otrzymane w Excelu.

Wykres:



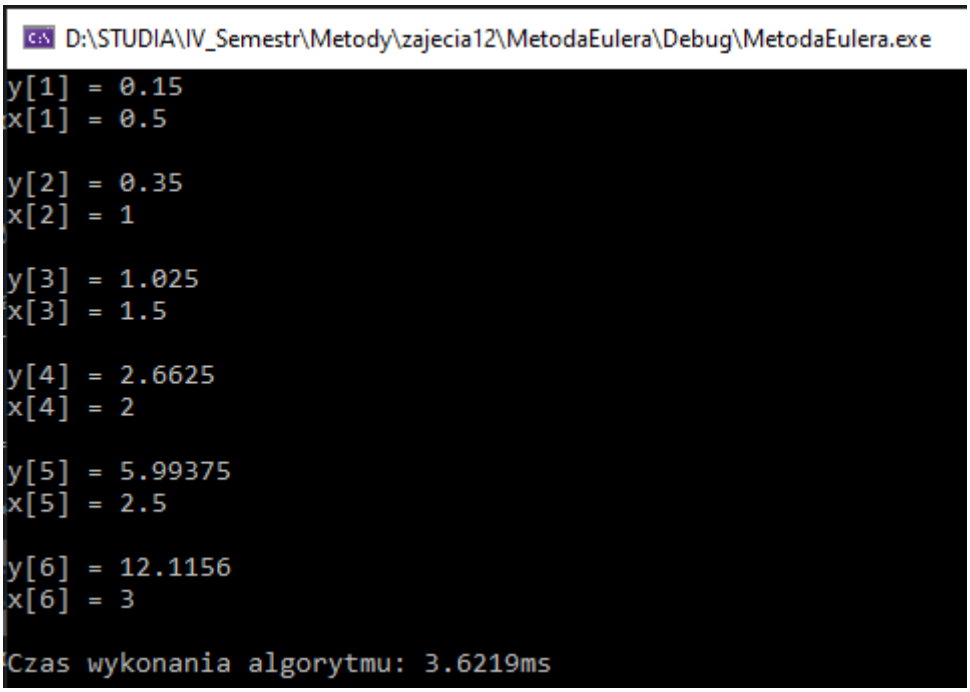
Rysunek 19.Wykres porównujący wartości wyliczone w programie z wartościami dokładnymi.

Przypadek C.

Tabela 6.Dane do test 2, przypadku C.

x_0	y_0	b	h	$\frac{dy}{dx}$	Rozwiązanie analityczne
0	0.1	3	0.5	$x^2 + y$	$2{,}1 * e^x - x^2 - 2x - 2$

Wartości zwrócone przez program:



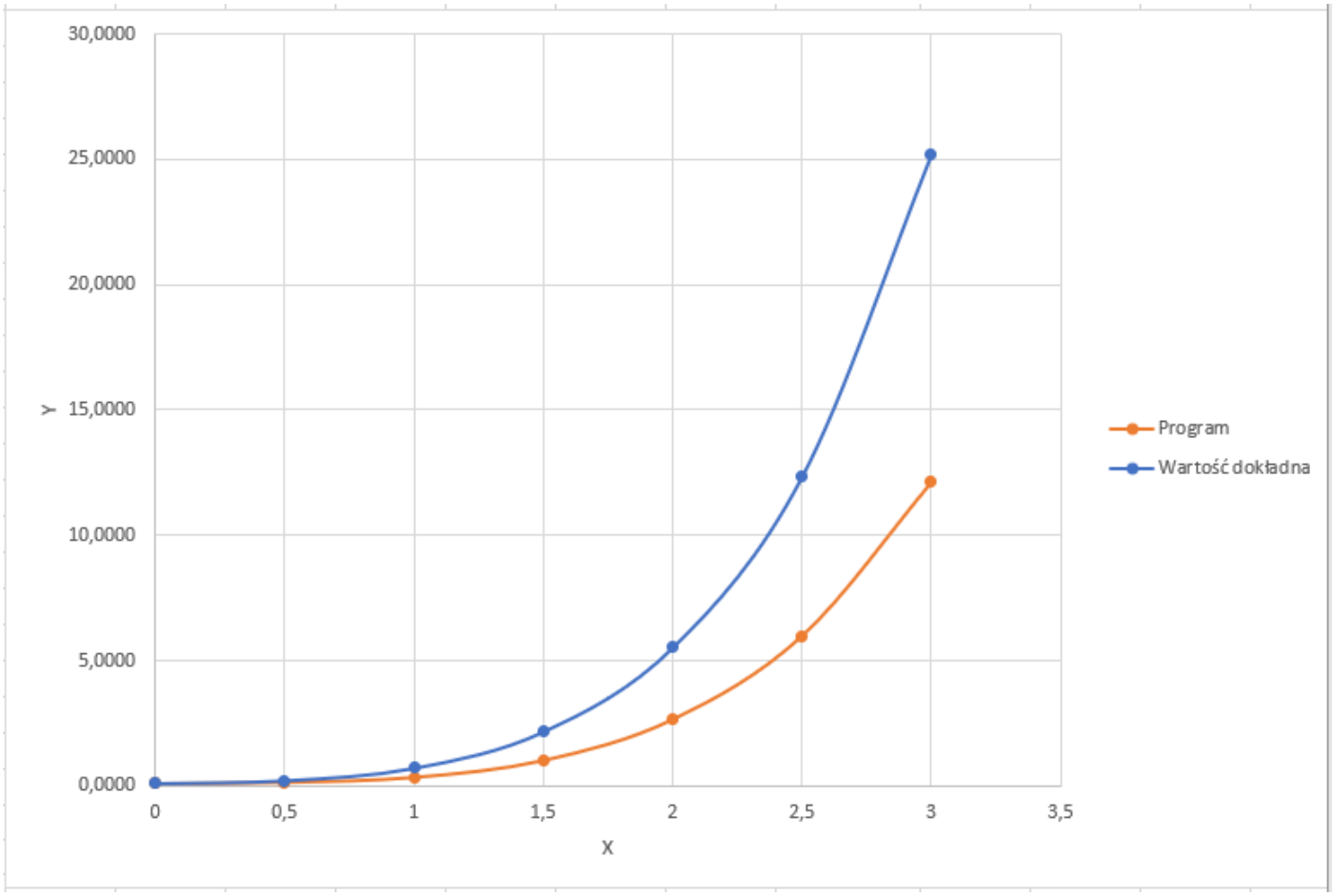
Rysunek 20.Wszystkie wartości zwrócone przez program dla h = 0.5.

Wartości obliczone w Excelu:

x0	0	y0	0,1000	y0	0,1
x1	0,5	y1	0,1500	y1	0,212314668
x2	1	y2	0,3500	y2	0,70839184
x3	1,5	y3	1,0250	y3	2,161547048
x4	2	y4	2,6625	y4	5,517017808
x5	2,5	y5	5,9938	y5	12,33323732
x6	3	y6	12,1156	y6	25,17962754
Punkty		Program		2,1*EXP(X) - x^2 -2x -2	

Rysunek 21.Wyniki otrzymane w Excelu.

Wykres:



Rysunek 22.Wykres porównujący wartości wyliczone w programie z wartościami dokładnymi.

5. Wnioski

Metoda Eulera, nazywana również metodą Rungego-Kutty (1 rzędu), jest najprostszą numeryczną metodą obliczania równania różniczkowego pierwszego rzędu. Niestety cecha ta jest przyczyną dosyć dużej niedokładności obliczeń i sporych błędów przy przybliżaniu szukanej funkcji. Analizując przeprowadzone testy łatwo można zauważyć, że precyzja tej metody silnie zależy od wielkości ustalonego kroku między kolejnymi punktami. Zmniejszając jego wielkość, czyli tym samy zwiększając liczbę punktów w przedziale jesteśmy w stanie dosyć dobrze redukować wielkości błędów podczas przybliżania, jednakże taka operacja jest kosztowna obliczeniowo, co z kolei przekłada się na znacznie dłuższy czas wykonania algorytmu. Efektywność tej metody nie jest zbyt duża i służy ona raczej jako podstawa dla bardziej złożonych metod.