Laboratorium 10

Artem Buhera
GĆ01 135678

16.05.2021

W danym laboratorium musieliśmy zaimplementować program do rozwiązywania równania różniczkowego zwyczajnego pierwszego rzędu

$$\frac{dy(t)}{dt} + \frac{10t^2+20}{t^2+1}(y(t)-1) = 0, \quad \text{dla } t \geq 0$$

z warunkiem początkowym $y(0) = 0$.

Używaliśmy trzech metod:

- bezpośredniej Eulera (BME)
- pośredniej Eulera (PME)
- metody trapezów (PMT)

BME

Na podstawie równania różnicowego $\frac{y_{k+1}-y_k}{\delta t} - f(t_k, y_k) = 0$ wyprowadzamy wzór na y_{k+1} :

$$\begin{aligned}
&\frac{y_{k+1} - y_k}{\delta t} - f(t_k, y_k) = 0 \;\Leftrightarrow \\
&\Leftrightarrow\; y_{k+1} - y_k = f(t_k, y_k)\delta t \;\Leftrightarrow \\
&\Leftrightarrow\; y_{k+1} = y_k + f(t_k, y_k)\delta t \;\Rightarrow \\
&\Rightarrow\; y_{k+1} = y_k + (1-y_k)\,\frac{10t_k^2 + 20}{t_k^2 + 1}\,\delta t
\end{aligned}$$

PME

Na podstawie równania różnicowego $\frac{y_{k+1}-y_k}{\delta t} - f(t_{k+1}, y_{k+1}) = 0$ wyprowadzamy wzór na y_{k+1} :

$$f_{temp}(t) = \frac{10t^2 + 20}{t^2 + 1}$$

$$\begin{aligned}
&\frac{y_{k+1}-y_k}{\delta t} - f(t_{k+1}, y_{k+1}) = 0 \;\Leftrightarrow \\
&\Leftrightarrow\; y_{k+1} = y_k + f(t_{k+1}, y_{k+1})\delta t \;\Leftrightarrow \\
&\Leftrightarrow\; y_{k+1} = y_k + f_{temp}(t_{k+1})\,(1-y_{k+1})\,\delta t \;\Leftrightarrow \\
&\Leftrightarrow\; y_{k+1} = y_k + f_{temp}(t_{k+1})\,\delta t - y_{k+1}\,f_{temp}(t_{k+1})\,\delta t \;\Leftrightarrow \\
&\Leftrightarrow\; y_{k+1}\,(1 + f_{temp}(t_{k+1})) = y_k + f_{temp}(t_{k+1})\,\delta t \;\Leftrightarrow \\
&\Leftrightarrow\; y_{k+1} = \frac{y_k + f_{temp}(t_{k+1})\,\delta t}{1 + f_{temp}(t_{k+1})\delta t}
\end{aligned}$$

PMT

Na podstawie równania różnicowego $\frac{y_{k+1}-y_k}{\delta t} - \frac{f(t_k, y_k) + f(t_{k+1}, y_{k+1})}{2} = 0$ wyprowadzamy wzór na y_{k+1} :

$$f_{temp}(t) = \frac{10t^2 + 20}{t^2 + 1}$$

$$\frac{y_{k+1}-y_k}{\delta t} - \frac{f(t_k, y_k) + f(t_{k+1}, y_{k+1})}{2} = 0 \Leftrightarrow$$

$$\Leftrightarrow \; y_{k+1}-y_k = \frac{1}{2}\delta t\,[f(t_k, y_k) + f(t_{k+1}, y_{k+1})] \Leftrightarrow$$

$$\Leftrightarrow \; y_{k+1}-y_k = \frac{1}{2}\delta t\,[f_{temp}(t_k)\,(1-y_k) + f_{temp}(t_{k+1})\,(1-y_{k+1})] \Leftrightarrow$$

$$\Leftrightarrow \; y_{k+1}-y_k = \frac{1}{2}\delta t\, f_{temp}(t_k) - \frac{1}{2}\delta t\; y_k\, f_{temp}(t_k) + \frac{1}{2}\delta t\, f_{temp}(t_{k+1}) - \frac{1}{2}\delta t\; y_{k+1}\, f_{temp}(t_{k+1}) \Leftrightarrow$$

$$\Leftrightarrow \; y_{k+1} + \frac{1}{2}\delta t\; y_{k+1}\, f_{temp}(t_{k+1}) = y_k + \frac{1}{2}\delta t\, f_{temp}(t_k) - \frac{1}{2}\delta t\; y_k\, f_{temp}(t_k) + \frac{1}{2}\delta t\, f_{temp}(t_{k+1}) \Leftrightarrow$$

$$\Leftrightarrow \; y_{k+1}\,[1 + \frac{1}{2}\delta t\, f_{temp}(t_{k+1})] = y_k + \frac{1}{2}\delta t\, f_{temp}(t_k) - \frac{1}{2}\delta t\; y_k\, f_{temp}(t_k) + \frac{1}{2}\delta t\, f_{temp}(t_{k+1}) \Leftrightarrow$$

$$\Leftrightarrow \; y_{k+1} = \frac{y_k + \frac{1}{2}\delta t\, f_{temp}(t_k) - \frac{1}{2}\delta t\; y_k\, f_{temp}(t_k) + \frac{1}{2}\delta t\, f_{temp}(t_{k+1})}{1 + \frac{1}{2}\delta t\, f_{temp}(t_{k+1})}$$

Kod programu:

```cpp
#include <cmath>
#include <fstream>
#include <iostream>

using namespace std;


double f_exact(double t) {
    return 1 - exp(-10*(t + atan(t)));
}

double solveMethod(double left_bound, double right_bound, double h,
                   double (*calculate_y_k1)(double t_k, double y_k, double h),
                   string filename, string header) {

    double y_k1 = 0.0, y_k = 0.0;
    double t_k = left_bound;
    double maxError = 0, currentError;

    ofstream out;
    bool shouldSave = (!filename.empty()) && !(header.empty());
    if (shouldSave) {
        out.open("../lab10/" + filename);
        out << header << endl;
    }

    while (t_k < right_bound) {
        y_k1 = calculate_y_k1(t_k, y_k, h);

        currentError = abs(y_k1 - f_exact(t_k + h));
        if (maxError < currentError) {
            maxError = currentError;
        }

        t_k += h;
        y_k = y_k1;

        if (shouldSave)
            out << t_k << ',' << y_k << endl;
    }

    out.close();

    return maxError;
}
```

```
double f_temp(double t){
    return (10.0 * t * t + 20.0) / (t * t + 1.0);
}


double calculate_y_k1_BME(double t_k, double y_k, double h) {
    return y_k + (1.0 - y_k) * f_temp(t_k) * h;
}

double calculate_y_k1_PME(double t_k, double y_k, double h) {
    double temp_k = f_temp(t_k);
    return (y_k + temp_k * h) / (1.0 + temp_k * h);
}

double calculate_y_k1_PMT(double t_k, double y_k, double h) {
    double temp_k = f_temp(t_k);
    double temp_k1 = f_temp(t_k + h);
    return (y_k + h / 2.0 * temp_k + h / 2.0 * temp_k1 - y_k * h / 2.0 * temp_k)
           / (h / 2.0 * temp_k1 + 1.0);
}


double solveBME(double left_bound, double right_bound, double h,
                string filename="", string header="s") {
    return solveMethod(left_bound, right_bound, h, calculate_y_k1_BME,
                       filename, header);
}

double solvePME(double left_bound, double right_bound, double h,
                string filename="", string header="") {
    return solveMethod(left_bound, right_bound, h, calculate_y_k1_PME,
                       filename, header);
}

double solvePMT(double left_bound, double right_bound, double h,
                string filename="", string header="") {
    return solveMethod(left_bound, right_bound, h, calculate_y_k1_PMT,
                       filename, header);
}


double order(double h_i, double h_j, double y_i, double y_j) {
    return (y_i - y_j) / (h_i - h_j);
}

void BME_PME_PMT_errors(double left_bound, double right_bound) {
    ofstream out;
    out.open("../lab10/BME_PME_PMT_errors.csv");
    out << "h,BME,PME,PMT" << endl;

    double error_BME, error_PME, error_PMT;
    double h_0,  BME_0,  PME_0,  PMT_0,
           h_10, BME_10, PME_10, PMT_10;
```

```
    int count = 0;
    double h = 0.001;
    while (h > 1e-10) {
        error_BME = solveBME(left_bound, right_bound, h);
        error_PME = solvePME(left_bound, right_bound, h);
        error_PMT = solvePMT(left_bound, right_bound, h);

        out << log10(h) << ','
            << log10(error_BME) << ','
            << log10(error_PME) << ','
            << log10(error_PMT) << endl;

        if (count == 0) {
            h_0 = log10(h);
            BME_0 = log10(error_BME);
            PME_0 = log10(error_PME);
            PMT_0 = log10(error_PMT);
        } else if (count == 10) {
            h_10 = log10(h);
            BME_10 = log10(error_BME);
            PME_10 = log10(error_PME);
            PMT_10 = log10(error_PMT);
        }

        h /= 2.0;
        count++;
    }

    cout << "rząd BME: " << order(h_0, h_10, BME_0, BME_10) << endl;
    cout << "rząd PME: " << order(h_0, h_10, PME_0, PME_10) << endl;
    cout << "rząd PMT: " << order(h_0, h_10, PMT_0, PMT_10) << endl;

    out.close();
}

int main() {
    double left_bound = 0.0, right_bound = 0.75;
      double h = 0.001, h_unstable = 0.25;

      solveBME(0.0, 3.0, h,
         "BME_stable.csv", "t,BME_stable h=0.01");
      solveBME(0.0, 3.0, h_unstable,
         "BME_unstable.csv", "t,BME_unstable h=0.25");
      solvePME(left_bound, right_bound, h,
         "PME.csv", "t,PME");
      solvePMT(left_bound, right_bound, h,
         "PMT.csv", "t,PMT");


    BME_PME_PMT_errors(left_bound, right_bound);
}
```

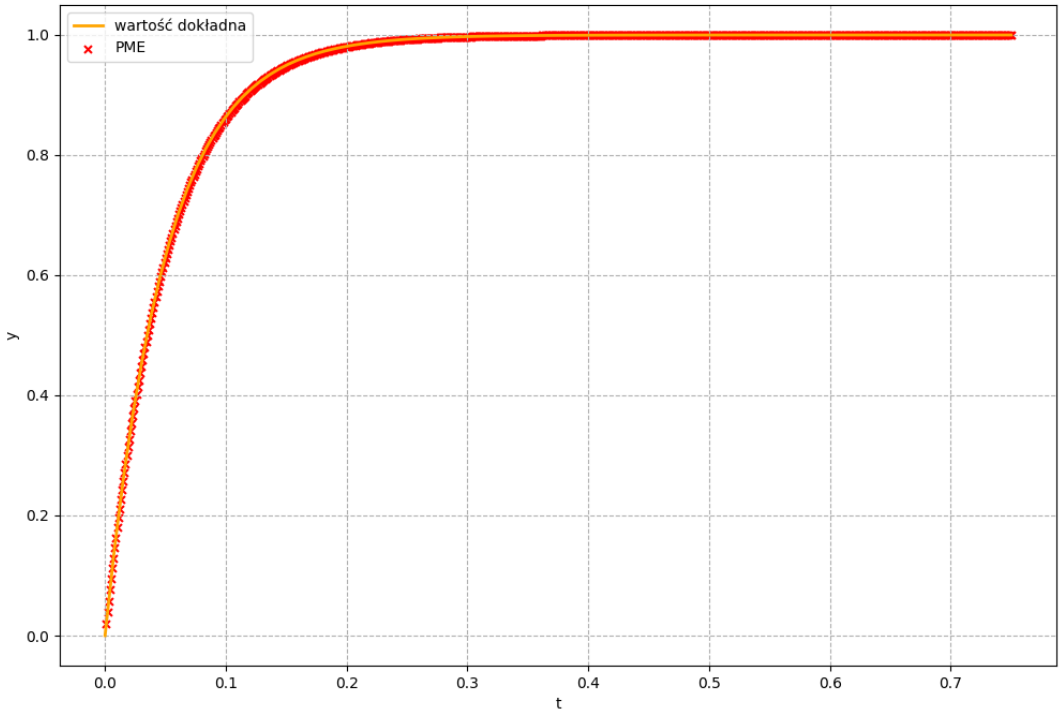
Wynik działania programu:

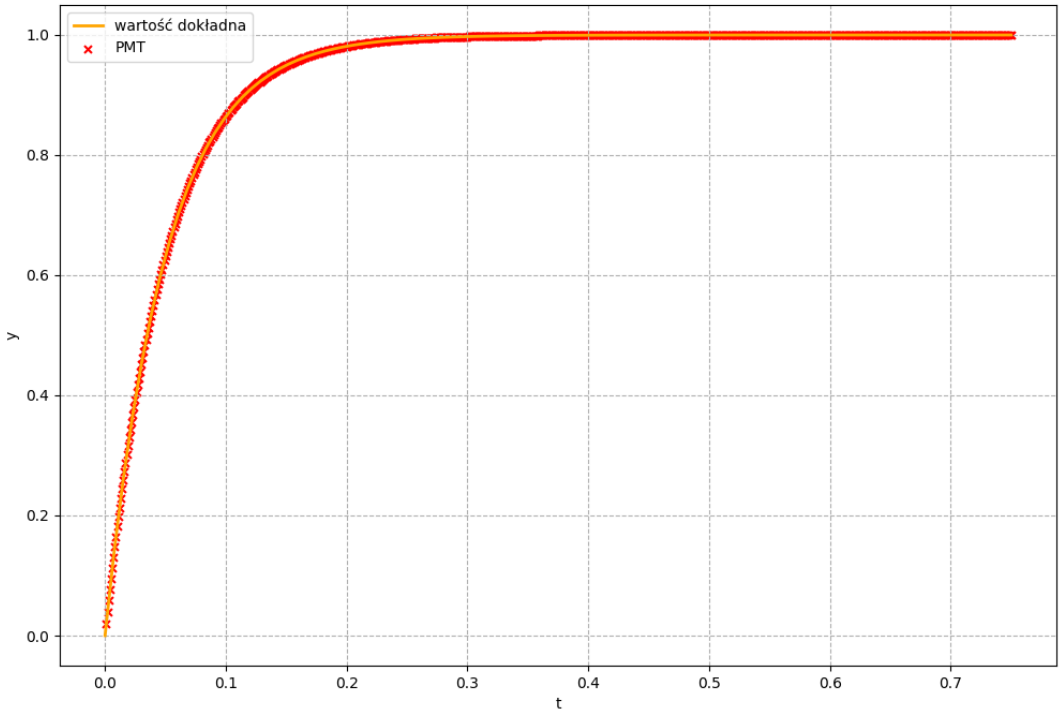
```
rząd BME: 1.00121
rząd PME: 0.998807
rząd PMT: 1.99733
```

Otrzymane wyniki rzędów dokładności zgadzają się z teoretycznymi:

- 1 dla BME
- 1 dla PME
- 2 dla PMT

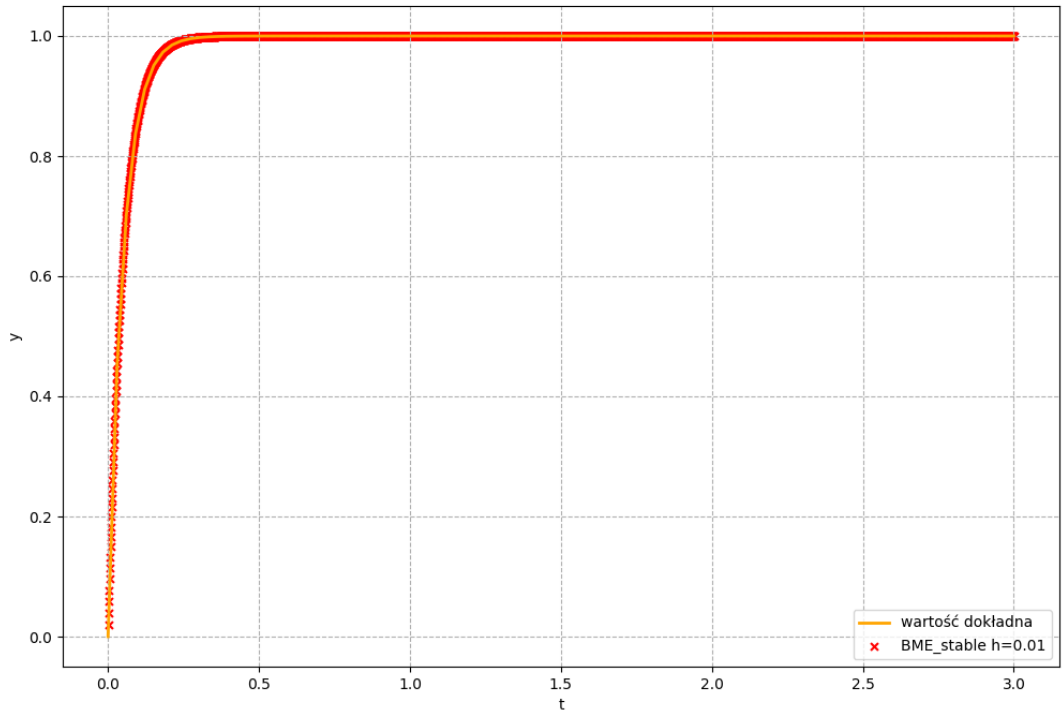
Otrzymane wykresy dla metod PME i PMT w porównaniu ze wzorem analitycznym $f_{exact}(t) = 1 - \exp[-10(t + \arctan(t))]$ dla kroku $h = 0.001$:

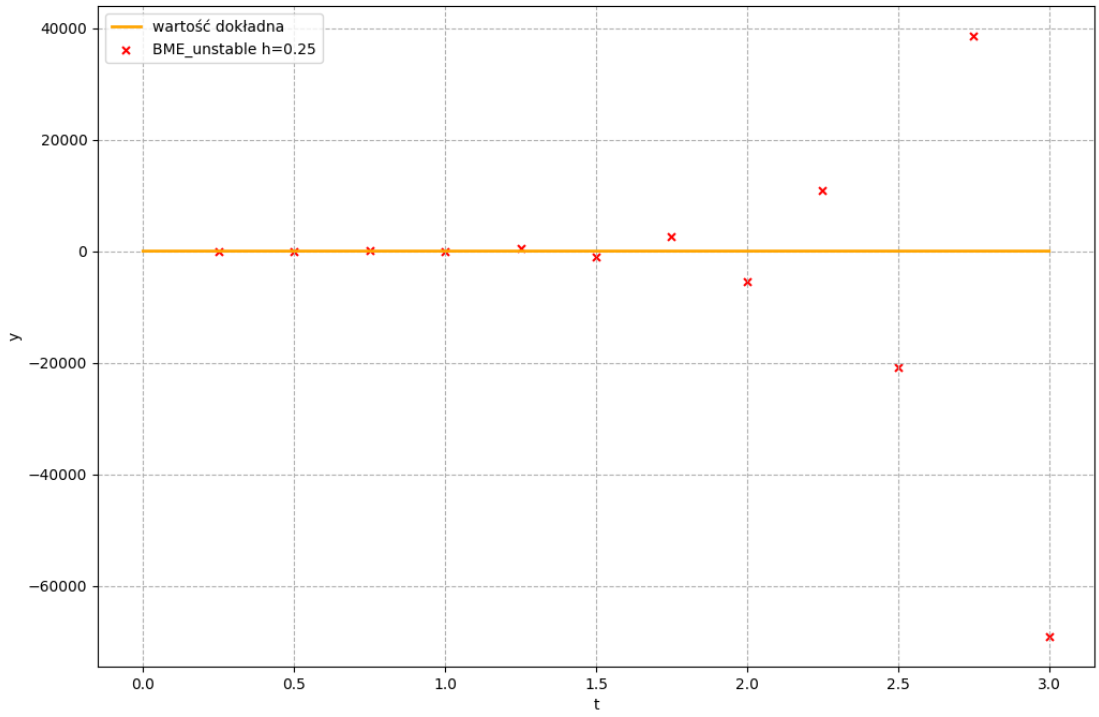

wartość dokładna
PME
1.0
0.8
0.6
0.4
0.2
0.0
y
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
t



Przy dosyć małym kroku otrzymaliśmy dobre wyniki dla obu metod

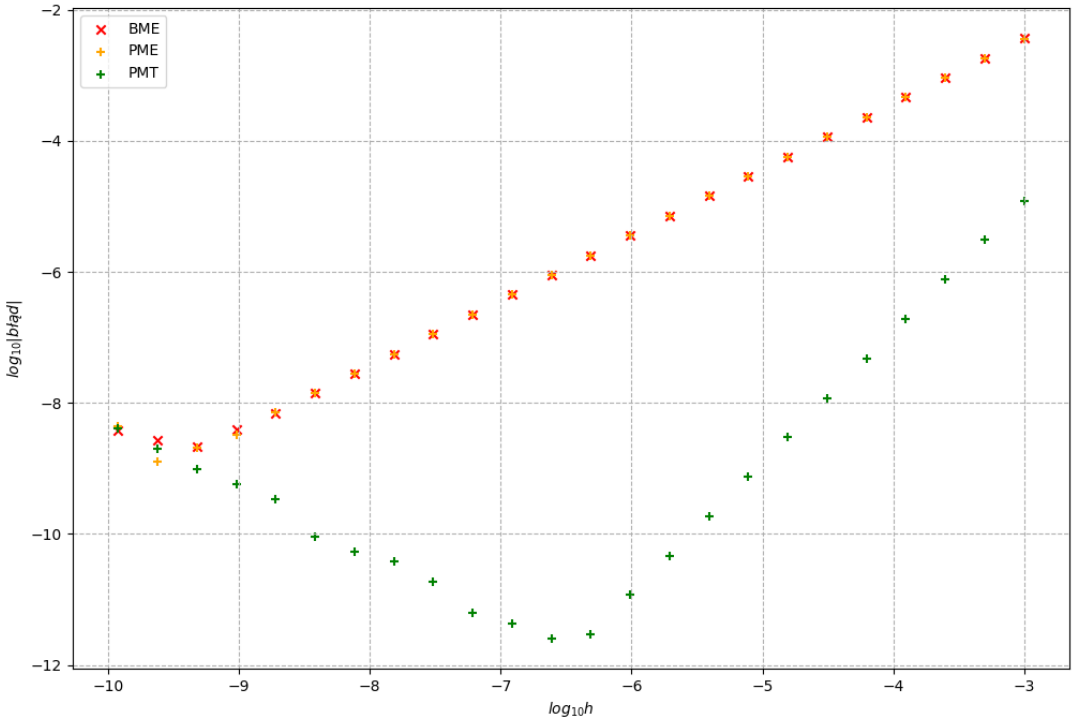
Otrzymane wykresy dla metody BME w porównaniu ze wzorem analitycznym dla kroku h = 0.001 oraz $h_{unstable}$ = 0.25, przy którym metoda jest numerycznie niestabilna:





Ewidentnie, dla zbyt dużego h metoda BME nie działa poprawnie.

Na wykresie zależności maksymalnego błędu bezwzględnego od kroku sieci *h* dla trzech metod możemy zaobserwować zgodność z teoretycznymi rzędami dokładności.



Co więcej, dla kroku $h \approx 10^{-6.5}$ dla metody PMT oraz $h \approx 10^{-9.5}$ odpowiednio dla BME i PME obserwujemy zmniejszenie dokładności obliczeń spowodowane błędami maszynowymi.