AGH

Program rozwiązujący równanie różniczkowe pierwszego rzędu metodą Midpoint

Paweł Gorgolewski

Wojciech Przybytek

Przemysław Roman

Spis treści

1. Wstęp teoretyczny

a. Rozwiązanie klasyczne

Rozważmy równanie różniczkowe liniowe pierwszego rzędu o następującej postaci

$$x'(t) = Ax(t) + Be^{-t}$$

gdzie A i B to pewne stałe rzeczywiste. Jak widać jest to równanie niejednorodne, posłużymy się więc do jego rozwiązania metodą uzmienniania stałej. Rozwiązujemy na początku równanie jednorodne:

$$x'(t) - Ax(t) = 0$$

$$x(t) = C_1 e^{At}$$

Następnie stałą C_1 zastępujemy pewną funkcją $u(t)$

$$x(t) = u(t)e^{At}$$

$$x'(t) = u'(t)e^{At} + u(t)Ae^{At}$$

i podstawiamy do pierwotnego równania

$$u'(t)e^{At} + u(t)Ae^{At} = Au(t)e^{At} + Be^{-t}$$

$$u'(t)e^{At} = Be^{-t}$$

$$u'(t) = Be^{-t-At}$$

Po scałkowaniu obu stron otrzymamy

$$u(t) = C - \frac{Be^{-t-At}}{A+1}$$

co po podstawieniu do rozwiązaniu równania jednorodnego daje nam ostateczne rozwiązanie

$$x(t) = Ce^{At} - \frac{Be^{-t}}{A+1}$$

Jeżeli wiemy dodatkowo, że dla pewnego argumentu t_a wartość funkcji wynosi x_a to możemy obliczyć wartość stałej C

$$C = \frac{x_a(A+1) + Be^{-t_a}}{e^{At_a}(A+1)}$$

b. Rozwiązanie numeryczne

Rozważmy ponownie równanie o postaci

$$x'(t) = Ax(t) + Be^{-t}$$

Pochodną można inaczej zapisać jako

$$x'(t) = f(t, x)$$

i przedstawić w formie granicy

$$\lim_{h \to 0} \frac{x(t+h) - x(t)}{h} = f(t, x)$$

Przybliżając wartość pochodnej do wartości ilorazu różnicowego otrzymamy

$$\frac{x(t+h) - x(t)}{h} = f(t, x)$$

Przyjmując, że $t_{i+1} = t_i + h$ otrzymujemy

$$x(t_{i+1}) = x(t_i) + hf(t_i, x(t_i))$$

czyli inaczej

$$x_{i+1} = x_i + hf(t_i, x_i)$$

W ten sposób znając warunek początkowy $x(t_a) = x_a$ oraz przyjmując wartość kroku h na przedziale $< a, b >$, można otrzymać numeryczne rozwiązanie równania różniczkowego, czyli zbiór punktów, które przybliżają funkcję $x(t)$. Taką metodę nazywa się metodą Eulera. Można ją jednak jeszcze udoskonalić i zamiast przybliżać wartość funkcji w przedziale $< t, t + h >$ do wartości na początku tego przedziału, obliczyć za pomocą metody Eulera wartość w połowie przedziału, a następnie za jej pomocą obliczyć wartość na jego końcu

$$t_m = t_i + \frac{h}{2}$$

$$x_m = x_i + \frac{h}{2} f(t_i, x_i)$$

$$x_{i+1} = x_i + hf(t_m, x_m)$$

Jest to modyfikacja metody Eulera nazywana metodą Midpoint i pozwala na otrzymanie dokładniejszych przybliżeń funkcji dla tego samego kroku niż standardowa metoda Eulera.

c. Porównanie rozwiązań

Przyjmując dla poprzednio przywołanego przykładu wartości stałych

$$A = -3,\ B = 2,\ t_a = 0,\ x_a = 2$$

otrzymamy równanie

$$x'(t) = -3x(t) + 2e^{-t}$$

którego rozwiązaniem po uproszczeniu jest

$$x(t) = e^{-3t} + e^{-t}$$

Obliczamy wartość funkcji dla $t = 0.1$ każdym z przedstawionych sposobów:

Wartość dokładna

$$x(0.1) = e^{-0.3} + e^{-0.1} \approx 1.645656$$

Metoda Eulera

$$x(0.1) \approx x_a + hf(x_a, x_t) = 2 + 0.1 \times (-3 \times 2 + 2 \times e^{0}) = 1.6$$

$$|1.645656 - 1.6| = 0.045656$$

Metoda Midpoint

$$t_m = t_i + \frac{h}{2} = 0.05$$

$$x_m = x_i + \frac{h}{2} f(t_a, x_a) = 2 + 0.05 \times (-3 \times 2 + 2 \times e^{0}) = 1.8$$

$$x(0.1) \approx x_a + hf(t_m, x_m) = 2 + 0.1 \times (-3 \times 1.8 + 2 \times e^{0.05}) \approx 1.67025$$

$$|1.645656 - 1.67025| = 0.024594$$

2. Opis programu

a. Instalacja

➔ Linux

Aby zainstalować zależności programu należy w terminalu przejść do lokalizacji, w której znajduje się pobrany kod i wywołać komendę `sudo ./install.sh` (należy do tego mieć uprawnienia sudo oraz zainstalowanego basha)

➔ Windows

Aby zainstalować zależności programu należy mieć na swoim urządzeniu zainstalowaną oficjalną dystrybucję Python 3 (Download Python). Następnie w wierszu poleceń należy przejść do lokalizacji, w której znajduje się pobrany kod i wywołać komendę `pip install -r requirements.txt`

b. Uruchomienie programu

Program uruchamiany jest z wiersza poleceń komendą `./run.sh` (Linux) lub `python midpoint.py` (Windows).
Program można uruchomić z parametrami, których listę i opis można zobaczyć po uruchomieniu programu z argumentem `-h` lub `--help`

```
usage: midpoint.py [-h] [-A A] [-B B] [-a A] [-b B] [-n N] [-xa
XA] [-E E]

Midpoint method for equations of the form x'(t) = A*x + B*exp(-t)

optional arguments:
  -h, --help  show this help message and exit
  -A A        A coefficient
  -B B        B coefficient
  -a A        Start of the range to be inspected
```

```
-b B        End of the range to be inspected
-n N        Amount of steps
-xa XA      f(ta), initial value
-E E        Precision of solution
```

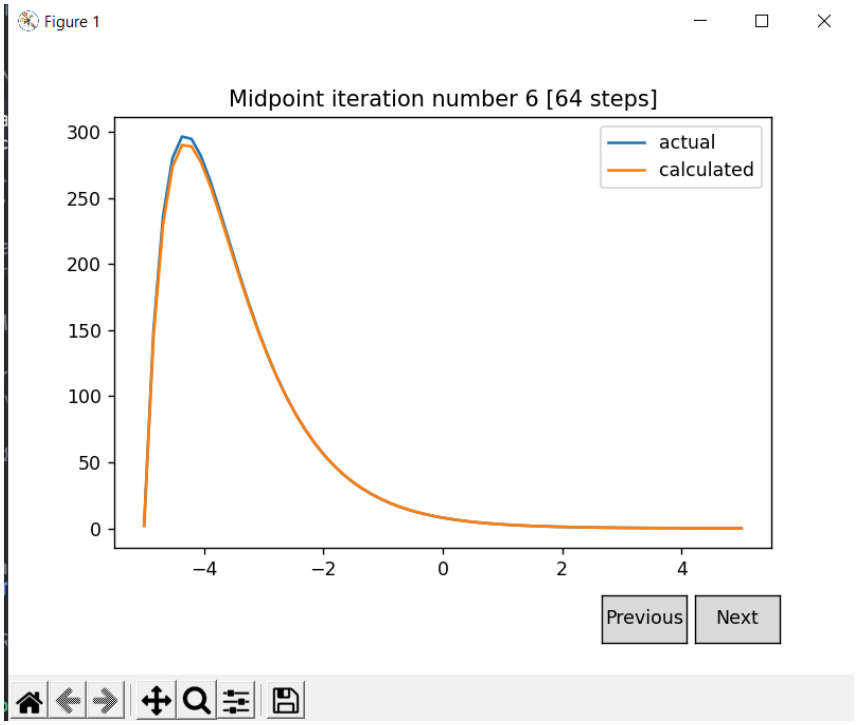
Domyślnie, przy uruchomieniu bez sprecyzowania argumentów, przyjmują one następujące wartości:

- $A = -3$
- $B = 2$
- $a = 0$
- $b = 10$
- $n = 16$
- $x_a = 2$
- $\varepsilon = 0.001$

Uwaga, ponieważ program nie rozważa różnych przypadków rozwiązania dla różnych wartości A i B, a zawsze oblicza wartość funkcji $x(t)$ z wyznaczonego ogólnego wzoru, niektóre wartości mogą zwracać złe wyniki. W szczególności z obserwacji wynika, że wartość A nie powinna być dodatnia ani wynosić -1.

Po uruchomieniu programu wyświetlą nam się dwa wykresy funkcji $x(t)$, dokładny oraz obliczony metodą Midpoint, zawierające wskazaną w parametrach wywołania liczbę punktów. Jeżeli aktualna iteracja nie spełni zadanej średniej dokładności, to program po jednej sekundzie podwoi liczbę kroków i ponowi obliczenia, do momentu aż nie zostanie spełniona średnia dokładność. Dodatkowo po każdej iteracji w konsoli pojawi się informacja dla każdego obliczonego punktu o jego wartości dokładnej i przybliżonej metodą Midpoint.

Po otrzymaniu satysfakcjonującej średniej dokładności program zatrzyma się i za pomocą przycisków *next* i *previous* można przeglądać wykresy wszystkich iteracji programu. Dodatkowo w katalogu w którym znajdował się program pojawi się plik w formacie .csv, który zawiera informacje o wszystkich obliczonych punktach dla każdej iteracji, wraz z ich wartością dokładną i przybliżoną metodą Midpoint.



okno programu po zakończeniu obliczeń

c. Opis działania programu

Program prezentuje działanie metody numerycznej Midpoint i porównuje wartości przez nią obliczone z wartościami dokładnymi, dla równania różniczkowego:

$$x'(t) = Ax(t) + Be^{-t},\ x(a) = x_a$$

na przedziale $< a,\ b >$. Obliczenia rozpoczynają się dla początkowych n punktów, ale z każdą iteracją, kiedy średnia dokładność metody nie wyniesie co najmniej ε, liczba punków jest podwajana.

➔ Wejście

Program do przetworzenia argumentów wywołania programu korzysta z wbudowanej biblioteki Pythona *argparse*. Tworzy obiekt klasy *ArgumentParser,* ustawia wszystkie możliwe argumenty, a po uruchomieniu sprawdza, czy użytkownik podał poprawne dane, to znaczy, czy zgadzają się typy danych, liczba kroków jest większa od 0, koniec przedziału jest większy niż jego początek oraz czy wprowadzone dane nie będą powodowały błędu programu.

➔ Obliczenia

Program w każdej iteracji oblicza wartości funkcji $x(t)$ dla n punktów. Wyniki dokładne i przybliżone przechowywane są w osobnych tablicach, a na koniec iteracji obliczana jest średnia różnica między wartościami w każdym punkcie i porównywana z oczekiwaną dokładnością. Odległość między kolejnymi punktami w przedziale jest stała dla danej iteracji i wynosi:

$$h = \frac{b-a}{n},$$

gdzie a - początek przedziału, b - koniec przedziału, n - liczba kroków

Wartość dokładna jest obliczana ze wzorów wyznaczonych w części Rozwiązanie klasyczne. Najpierw liczona jest stała C:

$$C = \frac{x_a(A+1) + Be^{-t_a}}{e^{A t_a}(A+1)},$$

gdzie jako t_a przyjmuje się początek obliczanego przedziału, a następnie wartości funkcji w punktach odległych o h w przedziale $< a, b >$:

$$x(t) = Ce^{At} - \frac{Be^{-t}}{A+1}$$

Wartość przybliżona jest obliczana zgodnie z metodą Midpoint omawianą w części Rozwiązanie numeryczne. Jako punkt początkowy przyjmuje się wartość funkcji na początku rozważanego przedziału, która to jest jednym z argumentów wywołania programu. Dla każdego kolejnego punktu oblicza się wartość korzystając ze wzorów:

$$t_m = t_i + \frac{h}{2}$$

$$x_m = x_i + \frac{h}{2} f(t_i, x_i)$$

$$x_{i+1} = x_i + hf(t_m, x_m)$$

Po wykonaniu obliczeń i otrzymaniu dwóch tablic program sumuje bezwzględną różnicę dla każdego punktu i dzieli ją przez ilość punktów otrzymując średnią dokładność metody, która to jest porównywana z parametrem ε. Jeżeli zadana wartość dokładności jest większa niż średnia obliczona, to program kończy obliczenia, w przeciwnym wypadku liczba kroków jest podwajana, a wszystkie poprzednie obliczenia są powtarzane.

➔ Wyjście

Program wyświetla się w oknie stworzonym dzięki bibliotece Matplotlib, na którym po każdej iteracji uaktualniane są wykresy funkcji $x(t)$ – wykres dokładny i przybliżony metodą Midpoint. Dodatkowo w konsoli są prezentowane dane dotyczące aktualnie obliczonej iteracji w formacie:

```
{'steps': [liczba kroków w iteracji],
'precision': [średnia dokładność iteracji],
'actual value': [rzeczywista wartość funkcji w punkcie],
'calculated value': [przybliżona wartość funkcji w punkcie]}
```

Te same dane są również zapisywane do pliku o nazwie *midpoint_report_[data].csv*, który tworzony jest w katalogu programu. Po zakończeniu wszystkich obliczeń okno programu pozostaje widoczne i za pomocą przycisków *next* i *previous* można przeglądać wykresy dla każdej iteracji.