Zastosowanie metody różnic skończonych do rozwiązania równania różniczkowego zwyczajnego rzędu drugiego

Dominik Adamczyk
Równania różniczkowe i różnicowe 2022/2023

Spis treści

1 Opis problemu

Przedstawiony w tym dokumencie problem polega na numerycznym rozwiązaniu równania różniczkowego (1) na przedziale $[0, 1]$

$$u''(x) - 2u'(x) + u(x) = x^2 + 3 \tag{1}$$

z zadanymi warunkami brzegowymi (2), (3)

$$u(0) = 1 \tag{2}$$

$$u(1) + 2u'(1) = 2 \tag{3}$$

Użyta do tego zostanie metoda różnic skończonych, umożliwiająca numeryczne obliczenie funkcji $u(x)$ przy pomocy jej przybliżonych pochodnych.

Dodatkowo w sprawozdaniu znajduje się wykres powyższej funkcji, wraz z porównaniem rozwiązania numerycznego z rozwiązaniem dokładnym.

2 Użyte narzędzia

Do implementacji procedur obliczających numeryczne rozwiązanie równania, a także do wykreślania odpowiednich grafów użyty został język programowania R, oraz środowisko RStudio. Sprawozdanie zostało napisane przy pomocy LaTeX z pakietem Knitr umożliwiającym wykonywanie kodu z języka R wewnątrz dokumentu.

3 Rozwiązanie dokładne

Pierwszym etapem powinno być sprawdzenie, czy powyższe równanie różniczkowe ma rozwiązanie. Wystarczy w tym celu obliczyć rozwiązanie jednorodne równania różniczkowego. W tym punkcie od razu zostanie wyznaczone dokładne równanie, gdyż jego wynik będzie potrzebny w dalszej części projektu.

Na początku wyznaczone zostanie rozwiązanie równania jednorodnego:

$$u''(x) - 2u'(x) + u(x) = 0$$

Równanie charakterystyczne w tym przypadku ma postać:

$$\Phi(\lambda) = \lambda^2 - 2\lambda + 1 = (\lambda - 1)^2$$

Powyższe równanie ma jeden podwójny pierwiastek rzeczywisty λ, więc funkcje $u_1(x) = e^x,\ u_2 = xe^x$ stanowią układ fundamentalny rozwiązania równania jednorodnego. Wtedy równanie jednorodne przyjmuje postać:

$$u_0(x) = c_1 e^x + c_2 x e^x \tag{4}$$

Kolejno wyznaczone zostanie równanie szczegółowe $U(x)$. Używając metody przewidywań przyjmuję:

$$\begin{aligned} U(x) &= a_2x^2 + a_1x + a_0 \\ U'(x) &= 2a_2x + a_1 \\ U''(x) &= 2a_2 \end{aligned}$$

Wstawiając powyższe równania do (1) otrzymujemy:

$$\begin{aligned} 2a_2 - 4a_2x - 2a_1 + a_2x^2 + a_1x + a_0 &= x^2 + 3 \\ a_2x^2 + (a_1 - 4a_2)x + a_0 - 2a_1 + a_2 &= x^2 + 3 \end{aligned}$$

Do wyznaczenia współczynników a_i konieczne będzie rozwiązanie układu równań:

$$\begin{cases} a_2 = 1 \\ a_1 - 4a_2 = 0 \\ a_0 - 2a_1 + 2a_2 = 3 \end{cases}$$

Po jego rozwiązaniu otrzymane zostaje rozwiązanie szczegółowe:

$$U(x) = x^2 + 4x + 9 \tag{5}$$

Rozwiązaniem równania (1) jest zatem suma rozwiązania jednorodnego i szczegółowego:

$$u(x) = u_0(x) + U(x) = c_1 e^x + c_2 x e^x + x^2 + 4x + 9 \tag{6}$$

Do wyznaczenia współczynnika c_1 wystarczy użyć warunku brzegowego (2). Po podstawieniu $x = 0$ otrzymamy:

$$c_1 + 9 = 1$$
$$c_1 = -8$$

Znając c_1 można obliczyć c_2 używając warunku brzegowego (3):

$$u(1) + 2u'(1) = -8e^1 + c_2 e + 1 + 4 + 9 + 2c_2 e + 2c_2 e - 16e + 4 + 8 = 2$$

$$c_2 = \frac{24 - 24e}{5e}$$

W ten sposób otrzymujemy algebraiczne rozwiązanie przedstawionego w punkcie pierwszym problemu początkowego:

$$u(x) = -8e^x + \frac{24 - 24e}{5e} x e^x + x^2 + 4x + 9 \tag{7}$$

4 Rozwiązanie numeryczne

4.1 Układ równań

Na potrzeby objaśnień użyta zostanie notacja $u(x_{i-1}), u(x_i), u(x_{i+1})$, gdzie $u(x_i)$ oznacza i-tą w kolejności wartość wyznaczanej funkcji $u(x)$. Algorytm będzie wyznaczał kolejne n wartości funkcji $u(x)$ w równych odstępach. Jeżeli za $h = \frac{1}{n}$ przyjmiemy odległość między dwoma kolejnymi wartościami funkcji $u(x)$, to $u(x_{i-1}) = u(x_i - h)$, oraz $u(x_{i+1}) = u(x_i + h)$. Wartości x_i wynoszą $i * h$, w szczególności $u(x_0) = u(0)$, a $u(x_n) = u(1)$.

Do wyznaczenia układu równań, na którego wynikiem będzie rozwiązanie równania w określonych punktach posłużą przybliżenia pierwszej i drugiej pochodnej:

$$u'(x_i) \approx \frac{u(x_{i+1}) - u(x_{i-1})}{h} \tag{8}$$

$$u''(x_i) \approx \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h} \tag{9}$$

Wstawiając powyższe przybliżenia pochodnych do równania (1) otrzymujemy:

$$\frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h} - 2\frac{u(x_{i+1}) - u(x_{i-1})}{h} + u(x_i) = x_i^2 + 3$$

co po przekształceniach daje:

$$u(x_{i-1})(\frac{1}{h^2} + \frac{1}{h}) + u(x_i)(1 - \frac{2}{h^2}) + u(x_{i+1})(\frac{1}{h^2} - \frac{1}{h}) = x_i^2 + 3 \tag{10}$$

Przy pomocy równania (10) konstruowany jest układ równań. Dla dla każdego $i \in \{2, .., n-1\}$ współczynniki przy $u(x_{i-1}), u(x_i), u(x_{i+1})$ będą takie same. Jedyne różnice występować będą dla $i = 1$, oraz $i = n$. Do ich rozwiązania konieczne będzie skorzystanie z warunków początkowych.

Dla $i = 1$ otrzymamy równanie:

$$u(x_0)(\frac{1}{h^2} + \frac{1}{h}) + u(x_1)(1 - \frac{2}{h^2}) + u(x_2)(\frac{1}{h^2} - \frac{1}{h}) = x_1^2 + 3$$

Jako że $u(x_0) = u(0)$, to możliwe jest skorzystanie z warunku brzegowego (2), w rezultacie otrzymując równanie:

$$u(x_1)(1 - \frac{2}{h^2}) + u(x_2)(\frac{1}{h^2} - \frac{1}{h}) = x_1^2 + 3 - (\frac{1}{h^2} + \frac{1}{h}) \tag{11}$$

Dla $i = n$ otrzymamy równanie:

$$u(x_{n-1})(\frac{1}{h^2} + \frac{1}{h}) + u(x_n)(1 - \frac{2}{h^2}) + u(x_{n+1})(\frac{1}{h^2} - \frac{1}{h}) = x_n^2 + 3 \tag{12}$$

Można je uprościć korzystając z równości $u(x_n) = u(1)$, warunku brzegowego (3) i przybliżenia pierwszej pochodnej (8). Przekształcając ten warunek otrzymamy:

$$u(x_n) + 2u'(x_n) = 2$$
$$u(x_n) + \frac{u(x_{n+1}) - u(x_{n-1})}{h} = 2$$
$$u(x_{n+1}) = 2h + u(x_{n-1}) - h * u(x_n)$$

Po wstawieniu wyliczonego powyżej $u(x_{n+1})$ do równania (12) otrzymamy:

$$u(x_{n-1})(\frac{1}{h^2} + \frac{1}{h}) + u(x_n)(1 - \frac{2}{h^2}) + (2h + u(x_{n-1}) - h * u(x_n))(\frac{1}{h^2} - \frac{1}{h}) = x_n^2 + 3$$

$$u(x_{n-1})(\frac{2}{h^2}) + u(x_n)(2 - \frac{1}{h} - \frac{2}{h^2}) = x_n^2 + 3 - 2h(\frac{1}{h^2} - \frac{1}{h}) \tag{13}$$

Przy pomocy równań (10), (11), (13) można stworzyć układ równań w postaci macierzowej $Ax = B$. Macierz A wypełniona zostanie współczynnikami po lewej stronie równości, macierz B to elementy po prawej stronie równości, a macierz x to szukane współczynniki $u(x_i)$. Pierwsze z wierszy macierzy A i B zostaną uzupełnione równością (11), ostatnie równością (13), a pozostałe przy pomocy równości (10).

4.2 Rozwiązywanie układu równań

Macierz A z poprzedniego układu równań jest szczególnym przypadkiem macierzy rzadkiej - macierzą trójdiagonalną. Równania, w których współczynniki reprezentowane są przez taką macierz możliwe są do rozwiązania w czasie liniowym $O(n)$, podczas gdy standardowa metoda Gaussa osiąga czas rzędu $O(n^3)$. Poniżej prezentowany jest algorytm Thomasa zaimplementowany w języku R służący do rozwiązania tego typu układów równań.

```r
tridiagonal_matrix <- function(below, diagonal, over, func) {
  # below - współczynniki tuż poniżej głównej przekątnej macierzy A
  # diagonal - współczynniki na głównej przekątnej macierzy A
  # over - współczynniki tuż nad główną przekątną macierzy A
  # func - wartości macierzy B
  n <- length(diagonal)

  over[1] <- over[1] / diagonal[1]
  func[1] <- func[1] / diagonal[1]

  for(i in 2:(n - 1)) {
    over[i] <- over[i] / (diagonal[i] - below[i - 1] * over[i - 1])
    func[i] <- (func[i] - below[i - 1] * func[i - 1]) /
      (diagonal[i] - below[i - 1] * over[i - 1])
  }
  func[n] <- (func[n] - below[n - 1] * func[n - 1]) /
    (diagonal[n] - below[n - 1] * over[n - 1])

  x <- vector(length = n)
  x[n] <- func[n]
  for(i in (n - 1):1)
    x[i] <- func[i] - over[i] * x[i + 1]

  return(x)
}
```

4.3 Algorytm główny

Główny algorytm programu sprowadza się do uzupełnienia w odpowiedni sposób wektorów reprezentujących trzy przekątne rozpatrywanej macierzy, a także wektora z wartościami macierzy B. Następnie uruchamiany jest algorytm Thomasa, który oblicza wartości funkcji $u(x)$.

```
f <- function(x){ # Prawa strona równania (10)
  return (x*x + 3)
}
```

```
solve_equation <- function(n){
  # n reprezentuje liczbę przedziałów na jakie zostanie podzielony obszar [0, 1]
  h = 1 / n

  # Inicjalizacja wektorów, nazwy analogiczne do tych z algorytmu Thomasa
  diag <- vector(length = n)
  below <- vector(length = n-1)
  over <- vector(length = n-1)
  func <- vector(length = n)

  # Współczynniki poszczególnych przekątnych wyznaczone w równaniu (10)
  diag_coof <- 1 - 2/h^2
  below_coof <- 1/h^2 + 1/h
  over_coof <- 1/h^2 - 1/h

  # Uzupełnianie przekątnch macierzy a i macierzy B
  for (i in 1:(n-1)){
    func[i] <- f(i * h)
    diag[i] <- diag_coof
    below[i] <- below_coof
    over[i] <- over_coof
  }

  # Uwzględnienie warunku początkowego dla pierwszego rzędu obliczonego w równaniu (11)
  func[1] = func[1] - below_coof

  # Uwzględnienie warunku początkowego dla ostatniego rzędu obliczonego w równaniu (13)
  diag[n] <- diag_coof + over_coof*(-h)
  below[n-1] <- below_coof + over_coof
  func[n] <- f(1) - over_coof * 2 * h

  output <- tridiagonal_matrix(below, diag, over, func)
  output <- c(1, output)
  return (output)
}
```

Funkcja $solve\_equation(n)$ rozwiązuje zadane równanie różniczkowe i zwraca wektor z $n + 1$ wartościami reprezentującymi kolejne wartości funkcji $u(x)$ na przedziale $[0, 1]$, tak, że i-ta wartość w wektorze reprezentuje wartość funkcji w punkcie $\frac{i}{n}$. Poniżej prezentowane są wyniki programu dla $n = 10$, oraz $n = 50$. Program działa dla dowolnego $n > 2$.

```
solve_equation(10)

##  [1] 1.0000000 0.9033881 0.8087136 0.7177924 0.6330244 0.5574966 0.4951015
##  [8] 0.4506730 0.4301419 0.4407133 0.4910705
```

```
solve_equation(50)

##  [1] 1.0000000 0.9806945 0.9614254 0.9422025 0.9230364 0.9039384 0.8849205
##  [8] 0.8659955 0.8471771 0.8284797 0.8099187 0.7915104 0.7732719 0.7552214
## [15] 0.7373779 0.7197617 0.7023938 0.6852967 0.6684936 0.6520093 0.6358694
## [22] 0.6201011 0.6047325 0.5897933 0.5753144 0.5613283 0.5478688 0.5349711
## [29] 0.5226722 0.5110103 0.5000258 0.4897602 0.4802571 0.4715617 0.4637213
## [36] 0.4567848 0.4508032 0.4458296 0.4419189 0.4391286 0.4375179 0.4371486
## [43] 0.4380847 0.4403928 0.4441416 0.4494028 0.4562503 0.4647611 0.4750148
## [50] 0.4870937 0.5010833
```

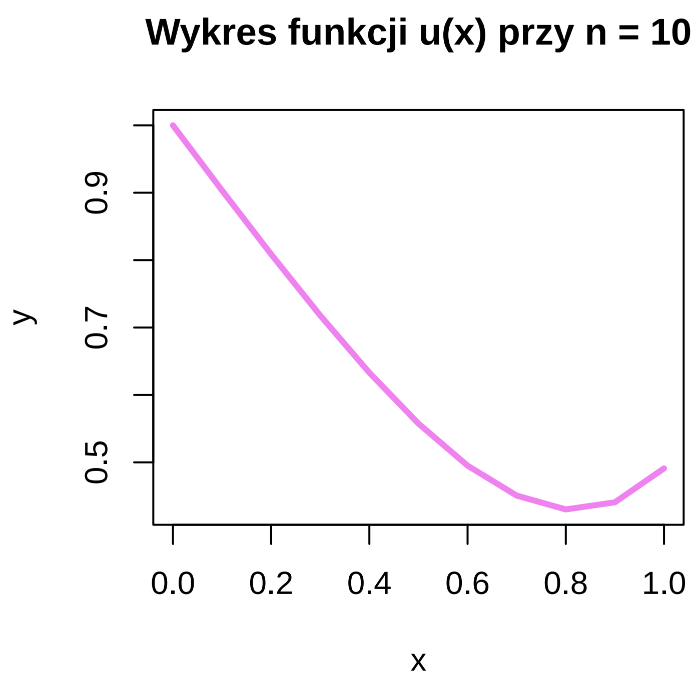
5 Wykres funkcji u(x)

Przedstawiona poniżej procedura odpowiada za narysowanie wykresu $u(x)$ przy pomocy wartości uzyskanych z funkcji *solve_equation*.

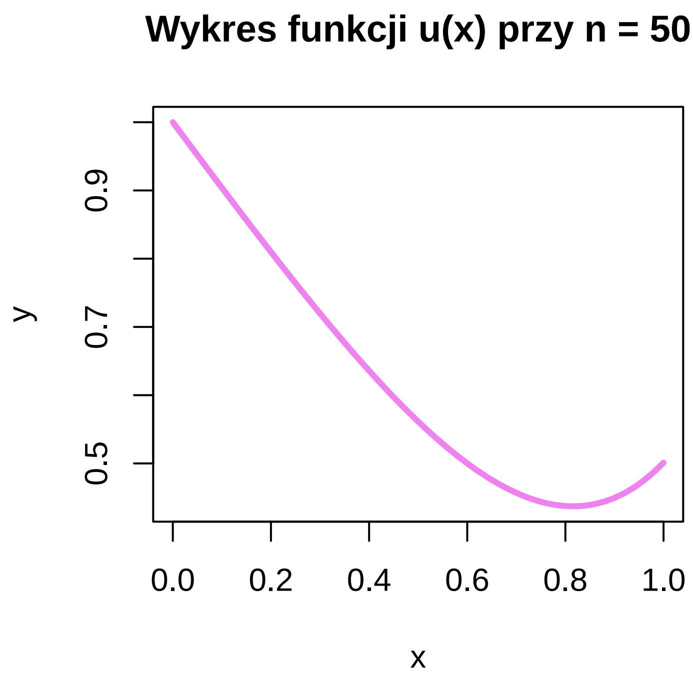
```
plot_approximate_solution <- function(eq, color="violet"){
  n = length(eq) - 1
  h = 1 / n
  a = 0
  b = 1
  x_vals = seq(a, b, h)

  plot(x_vals, eq, type="l", col=color, lwd="3",
       main=paste("Wykres funkcji u(x) przy n =", n), xlab="x", ylab="y")
}
```

Poniżej prezentowane są wykresy dla rozwiązań z wartościami $n = 10$, oraz $n = 50$.

```
plot_approximate_solution(solve_equation(10))
```



```
plot_approximate_solution(solve_equation(50))
```



6 Porównanie z rozwiązaniem dokładnym

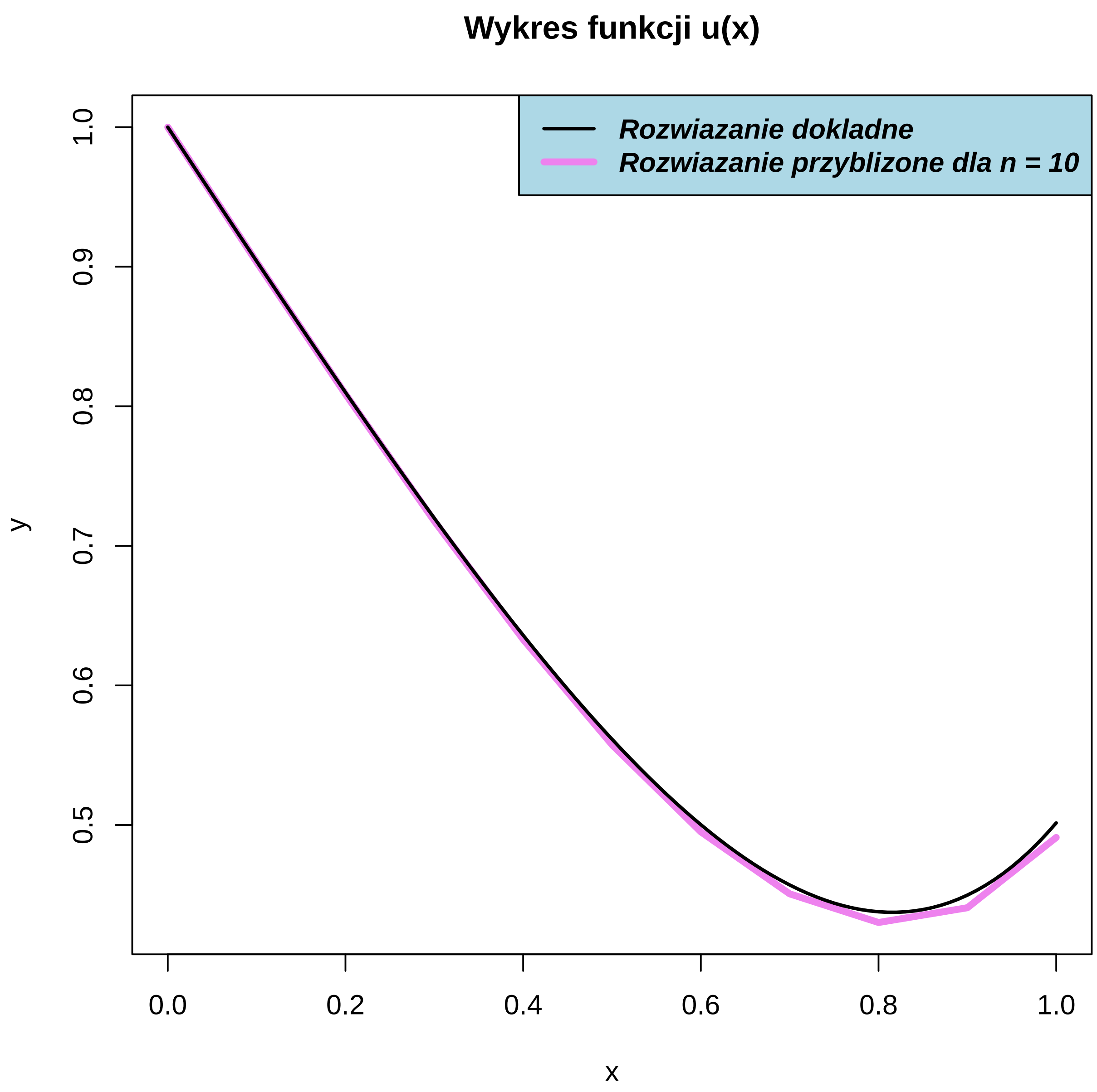
W punkcie 3 obliczone zostało rozwiązanie dokładne. Na jego podstawie wykonane zostało porównanie wyników zwracanych przez wcześniej opisane funkcje z rzeczywistymi wartościami funkcji $u(x)$. Poniżej przedstawiona funkcja *compare_with_exact* przyjmuje rozwiązanie przybliżone i porównuje je z rozwiązaniem dokładnym (obliczany przy pomocy funkcji *exact_u*. Porównanie polega na narysowaniu odpowiedniego wykresu, a także zwróceniu błędu globalnego.

```
exact_u <- function(x){
  (24*(exp(1) - 1)/(5 * exp(1)))*x*exp(x) - 8*exp(x) + x^2 + 4*x + 9
}

compare_with_exact <- function(approximation, color="violet"){
  n = length(approximation) - 1
  h = 1 / n
  a = 0
  b = 1
  x_vals = seq(a, b, h)

  plot(x_vals, approximation, type="l", col=color, lwd="4",
       main="Wykres funkcji u(x)", xlab="x", ylab="y")
  curve(exact_u, from=a, to=b, lwd=2, add=TRUE)
  legend(x="topright", legend=c("Rozwiazanie dokladne",
  paste("Rozwiazanie przyblizone dla n =",n)),
         col=c("black", "violet"), lwd=c(2, 4), text.font=4, bg='lightblue')

  exact_vals = exact_u(x_vals)
  error = abs(exact_vals - approximation)
  global_error <- max(error)
```

```
  return(global_error)
}
```

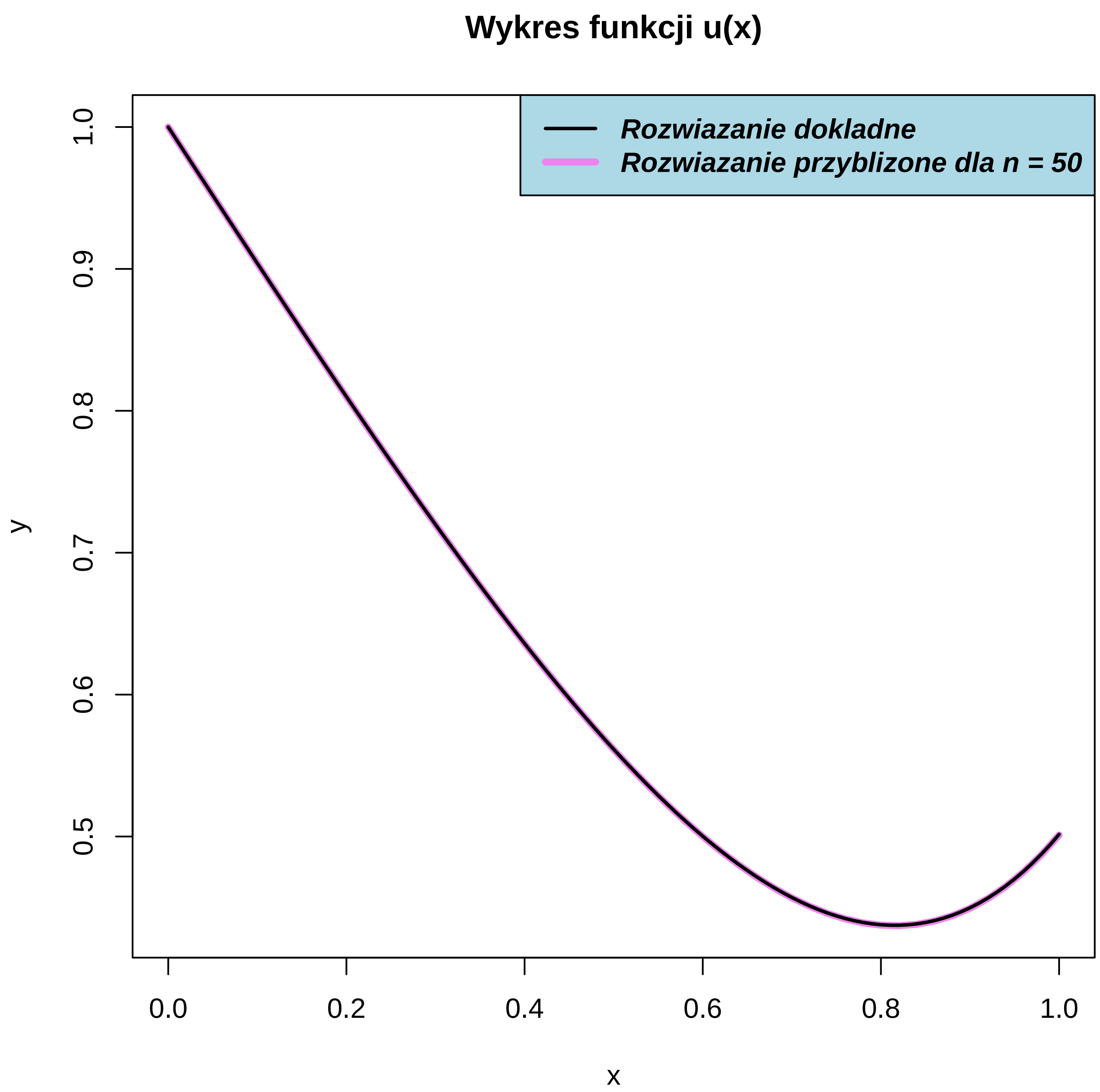
Poniżej prezentowane są porównania rozwiązania dokładnego i numerycznego funkcji dla wartości $n = 10$ i $n = 50$.

```
compare_with_exact(solve_equation(10))
```




```
## [1] 0.01042767

##      ^^^ błąd globalny
```

```
compare_with_exact(solve_equation(50))
```




```
## [1] 0.0004148776

##     ^^^ błąd globalny
```

7 Wnioski

Prezentowana w tym dokumencie metoda numerycznego obliczania wyniku równania różniczkowego pozwala z dobrą dokładnością wyznaczyć prawidłowe rozwiązanie, co pokazuje porównanie przedstawione w poprzednim punkcie. Dodatkowo wyznaczenie błędu globalnego pokazuje, że prezentowana metoda w istocie ma jest rzędu $O(h^2)$. Dla $n = 10$ (czyli $h = 0.1$) błąd globalny wynosił 0.01042767, czyli w przybliżeniu h^2, tak samo sytuacja wyglądała dla $n = 50$, czyli $h = 0.02$, gdzie błąd globalny wyniósł 0.0004148776. Prezentowana metoda dodatkowo ma bardzo dobrą złożoność obliczeniową rzędu $O(n)$ dzięki liniowemu rozwiązywaniu układu równań z użyciem algorytmu Thomasa.

Literatura

[1] https://en.wikipedia.org/wiki/Tridiagonal_matrix_algorithm