Akademia Górniczo-Hutnicza
Wydział Informatyki
Kierunek Informatyka

Metody Obliczeniowe w Nauce i Technice

Laboratorium 4

Efekt Rungego

Wojciech Michaluk, Kyrylo Iakymenko

Kraków, 22 marca 2024

1 Wprowadzenie

Podczas tego laboratorium zrealizujemy zadanie, które ma na celu pokazanie efektu Rungego - jest to zjawisko polegające na pogorszeniu jakości interpolacji wielomianowej, mimo zwiększenia liczby węzłów ([1], [2]), widoczne szczególnie na krańcach przedziału, zaobserwowane przez Carla Rungego w 1901r. Przeanalizujemy w tym celu wyniki interpolacji dwóch funkcji, porównując odmienne podejścia - z użyciem wielomianów Lagrange'a (warianty dla różnego rozmieszczenia węzłów) oraz funkcji sklejanych, tzw. *spline'ów*, a także badając rezultaty interpolacji w zależności od liczby węzłów. Następnie na podstawie uzyskanych wyników przedstawimy wnioski dotyczące tego zjawiska i różnych metod interpolacji.

2 Opis zadania

Celem zadania jest wyznaczenie wielomianów interpolujących dwie funkcje:

1. $f_1(x) = \frac{1}{1+25x^2}, x \in [-1, 1]$ - tzw. funkcja Rungego, tu szczególnie widoczny jest efekt Rungego;
2. $f_2(x) = e^{cos(x)}, x \in [0, 2\pi]$.

Użyjemy w tym celu trzech metod interpolacji, mianowicie:

- Interpolacja wielomianami Lagrange'a z równoodległymi węzłami, czyli $x_j = x_0 + jh$, gdzie j przyjmuje wartości $0, 1, ..., n$ oraz $h = \frac{x_n - x_0}{n}$.
- Interpolacja kubicznymi (sześciennymi) funkcjami sklejanymi, tzw. *cubic splines*, z równoodległymi węzłami (tak jak powyżej).
- Interpolacja wielomianami Lagrange'a, ale z węzłami Czebyszewa. Są one wyrażone wzorem $x_j = cos(\theta_j)$, przy czym $\theta_j = \frac{2j+1}{2(n+1)}\pi$ dla $0 \leq j \leq n$. Węzły te znajdują się w przedziale $[-1, 1]$. Dla funkcji f_2, aby węzły Czebyszewa obejmowały dziedzinę $[0, 2\pi]$, wykorzystujemy transformację zgodnie ze wzorem $x = a + (b - a) \cdot \frac{x_j + 1}{2}$ dla $x_j \in [-1, 1]$, podstawiając odpowiednio $a = 0$, $b = 2\pi$ i uzyskując $x = \pi \cdot (x_j + 1)$.

Porównamy funkcje interpolujące uzyskane w wyniku każdej z tych metod dla funkcji Rungego, a także zbadamy, jak zmieniają się rezultaty interpolacji tymi metodami w zależności od liczby węzłów - dla obu funkcji. Zanim przedstawiamy szerzej te zagadnienia, chcemy omówić specyfikę spline'ów.

2.1 Reprezentacja funkcji sklejanej i warunki, które musi ona spełniać

Funkcja sklejana $S(x)$ jest (jak sama nazwa mówi) "sklejona" z innych funkcji. Można to zapisać jako

$$S(x) = S_i(x) \text{ dla } x \in [t_i, t_{i+1}], \tag{1}$$

gdzie t_i oznacza węzły interpolacji, a S_i to funkcje, z których sklejamy.

W naszym podejściu używamy *cubic splines*, dla których muszą być spełnione następujące warunki:

1. $S_i(t_i) = y_i$ oraz $S_i(t_{i+1}) = y_{i+1}$ dla $i = 0, 1, ..., n-1$ | warunek interpolacji
2. $S'_{i-1}(t_i) = S'_i(t_i)$ dla $i = 1, 2, ..., n-1$ | ciągłość pierwszej pochodnej
3. $S''_{i-1}(t_i) = S''_i(t_i)$ dla $i = 1, 2, ..., n-1$ | ciągłość drugiej pochodnej

Na ich podstawie należy ułożyć odpowiedni układ równań, natomiast trzeba doprecyzować tzw. warunki brzegowe dla S_0 i S_{n-1}, co można zrobić m. in. przyjmując następujące:

- *natural spline* - $S_0''(t_0) = S_{n-1}''(t_n) = 0$,
- *clamped spline* - $S_0'(t_0) = f'(t_0)$ oraz $S_{n-1}'(t_n) = f'(t_n)$, gdzie f' jest pochodną interpolowanej funkcji,
- *not-a-knot spline* - $S_0'''(t_1) = S_1'''(t_1)$ oraz $S_{n-2}(t_{n-1}) = S_{n-1}(t_{n-1})$,
- *periodic spline* - $S_0'(t_0) = S_{n-1}'(t_n)$ oraz $S_0''(t_0) = S_{n-1}''(t_n)$,
- *quadratic spline* - S_0 i S_{n-1} są kwadratowe.

3 Interpolacja dla funkcji Rungego

W tej części zadania skupimy się na wyznaczeniu i analizie wielomianów interpolacyjnych i spline'a dla funkcji Rungego (f_1), przyjmując $n = 12$ węzłów interpolacji (choć zgodnie ze wzorem z opisu zadania, tak naprawdę liczba węzłów jest o 1 większa - przyjmujemy właśnie taki model). Następnie, aby próbkować wartości funkcji (interpolowanej oraz interpolujących) przyjmiemy dziesięciokrotne większe zagęszczenie węzłów w celu stworzenia wykresu tych funkcji. Najpierw przedstawiamy kody funkcji służących do generowania węzłów równoodległych i węzłów Czebyszewa, których używamy także do próbkowania (wtedy po prostu obsługujemy jedynie pierwszą zwróconą wartość).

```
def generate_n_equidistant(n, fun):
        j = np.arange(0, n + 1, 1)

        if fun == f1: #f1 to funkcja Rungego - wymieniona wyżej
                x_0, x_n = -1.0, 1.0
        elif fun == f2: #f2 analogicznie
                x_0, x_n = 0.0, 2 * np.pi

        h = (x_n - x_0) / n
        x = x_0 + j * h
        y = fun(x)
        return x, y
```

Listing 1: Funkcja generująca równoodległe węzły

```
def generate_n_chebyshev(n, fun):
        j = np.arange(0, n + 1, 1)
        theta = (2 * j + 1) / (2 * (n + 1)) * np.pi
        r = np.cos(theta)

        if fun == f1:
                x = r
        elif fun == f2:
                x = np.pi * (r + 1) #stosujemy transformację

        y = fun(x)
        return x, y
```

Listing 2: Funkcja generująca węzły Czebyszewa

Korzystając z tych funkcji, możemy przejść do wyznaczania wielomianów interpolacyjnych Lagrange'a oraz funkcji sklejanej. W tym drugim przypadku, korzystamy z bibliotecznej funkcji `scipy.interpolate.interp1d`, zaproponowanej w poleceniu zadania, która jednakże nie daje możliwości wyboru warunków brzegowych, natomiast próbuje dobrać jak "najgładszą" funkcję. Postanowiliśmy, że daje ona satysfakcjonujący rezultat,

natomiast można użyć także funkcji `scipy.interpolate.CubicSpline`, która przyjmuje jako jeden z parametrów właśnie rodzaj warunków brzegowych [1].

Z kolei w przypadku interpolacji Lagrange'a korzystamy z własnoręcznej implementacji, gdyż funkcja biblioteczna `scipy.interpolate.lagrange` jest niestabilna numerycznie (czyli podobnie jak w poprzednim laboratorium). Kod przedstawiamy poniżej.

```
def lagrange_interpolation_values(x, x_nodes, y_nodes):
        n = len(x_nodes)
        result = np.zeros(len(x))
        l_coeffs_denom = np.zeros(n)

        for i in range(n):
                l_coeffs_denom[i] = np.prod(x_nodes[:i] - x_nodes[i])
                  * np.prod(x_nodes[i + 1:] - x_nodes[i])

        for i in range(n):
                term = y_nodes[i]
                for j in range(n):
                        if j != i:
                                term *= (x - x_nodes[j])
                result += term / l_coeffs_denom[i]

        return result
```

Listing 3: Implementacja interpolacji Lagrange'a

Następnie generujemy węzły równoodległe i węzły Czebyszewa, dokonujemy próbkowania i wyznaczamy wielomiany interpolacyjne. W kodzie wygląda to następująco:

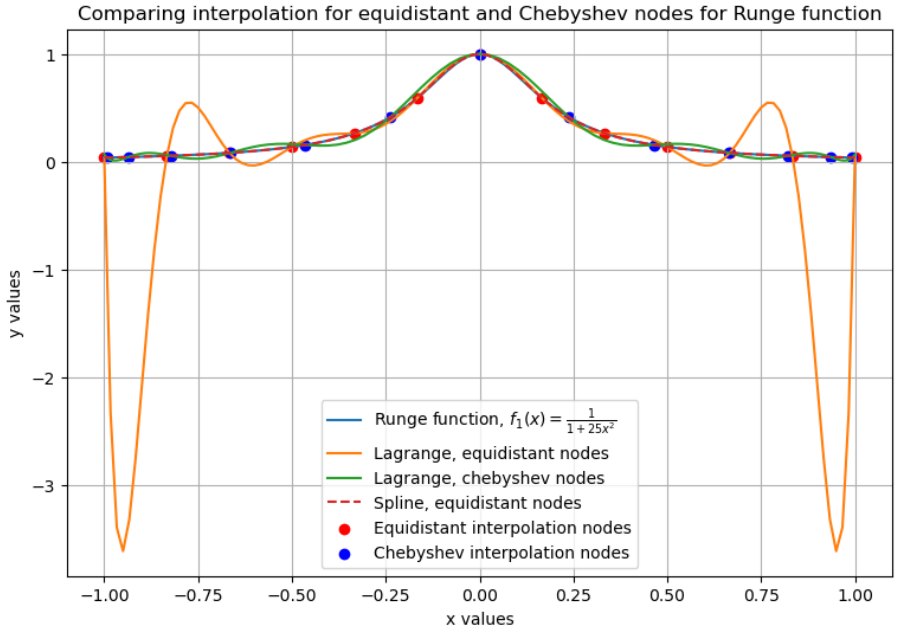
```
n = 12 #liczba węzłów

#generowanie węzłów - odpowiednio: równoodległych i Czebyszewa
x_nodes_equidistant, y_nodes_equidistant = generate_n_equidistant(n, f1)
x_nodes_chebyshev, y_nodes_chebyshev = generate_n_chebyshev(n, f1)

#próbkowanie
x_sample_equidistant = generate_n_equidistant(10 * n, f1)[0]
x_sample_chebyshev = generate_n_chebyshev(10 * n, f1)[0]

#funkcja sklejana i interpolacja z jej użyciem
spline_fun = spline.interp1d(x_nodes_equidistant,
  y_nodes_equidistant, kind = 'cubic')
spline_equidistant = spline_fun(x_sample_equidistant)

#interpolacja wielomianami Lagrange'a
lagrange_equidistant = lagrange_interpolation_values(x_sample_equidistant,
  x_nodes_equidistant, y_nodes_equidistant)
lagrange_chebyshev = lagrange_interpolation_values(x_sample_chebyshev,
  x_nodes_chebyshev, y_nodes_chebyshev)

y_values = f1(x_sample_equidistant) #obliczenie wartości funkcji
```

Listing 4: Wygenerowanie węzłów interpolacji i obliczenie funkcji interpolacyjnych

Teraz możemy przejść do narysowania wspólnego wykresu. Przygotujemy wykresy w dwóch wariantach: najpierw wszystko na jednym, wspólnym wykresie - jednakże to może być nieczytelne z powodu dużej ilości uwzględnianych do przedstawienia funkcji. Dlatego w drugim wariancie rozdzielamy funkcje i węzły na część związaną z równoodległymi węzłami i analogicznie na tę związaną z węzłami Czebyszewa (ale funkcję Rungego, czyli f_1, umieszczamy na obu).

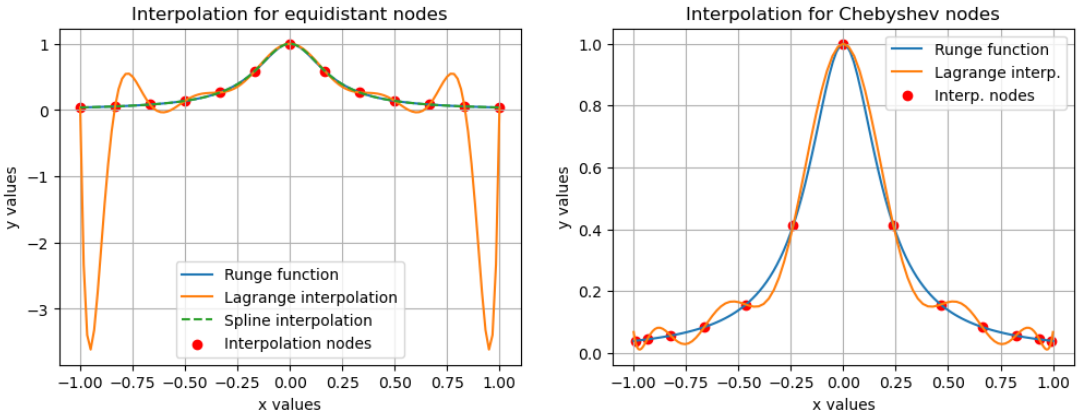
[1] Choć czy to daje pożądany efekt? Warto zajrzeć: `https://github.com/scipy/scipy/issues/14472`

Poniżej prezentujemy wspólny wykres ze wszystkimi funkcjami i węzłami.



Rysunek 1: Wykres wspólny wszystkich funkcji z zaznaczonymi węzłami interpolacji

Teraz obok siebie umieszczamy wykresy "rozdzielone" - po lewej stronie część poświęcona węzłom równoodległym, a po prawej - Czebyszewa.



Rysunek 2: Rozdzielone wykresy ze względu na rodzaje węzłów

4 Wyniki interpolacji w zależności od liczby węzłów

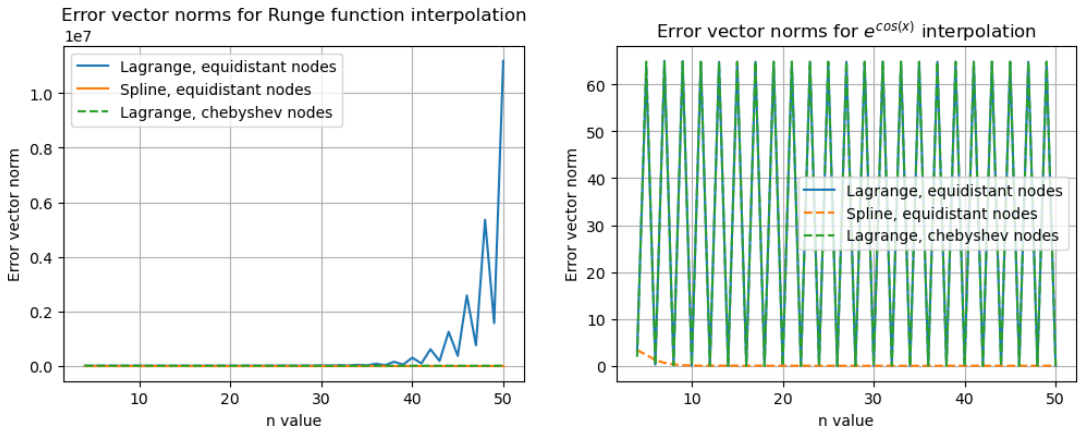
Spróbujemy teraz oszacować dokładność interpolacji we wszystkich 3 przypadkach. W tym celu porównamy wykresy funkcji norm wektorów błędów naszych metod interpolacji dla funkcji f_1 i f_2. W kodzie do realizacji tego celu używamy funkcji `np.linalg.norm` w następujący sposób:

```
def calculate_err_vec(fun_vals, approx_vals):
        #fun_vals to wartości funkcji, approx_vals to obliczone wartości
        return np.linalg.norm(np.abs(fun_vals - approx_vals))
```

Listing 5: Obliczanie normy wektora błędów

Obliczenia wykonujemy dla 500 losowo wybranych punktów z dziedziny dla obu funkcji, natomiast ustalamy ziarno (*seed*) losowania, aby uzyskać powtarzalne wyniki.

Poniżej prezentujemy wykres zależności normy wektora od ilości węzłów interpolacji n.

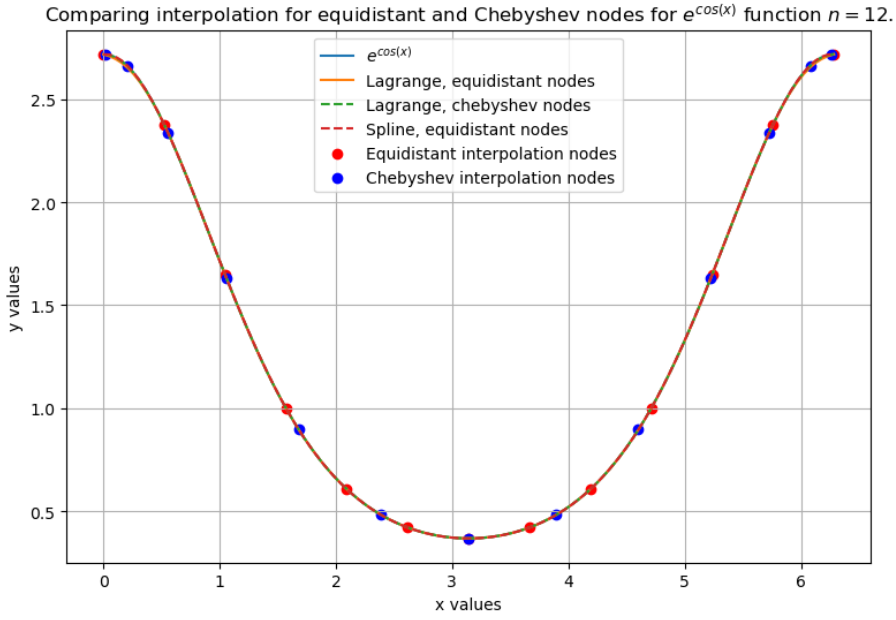


Rysunek 3: Wykres zależności normy wektora błędu od ilości węzłów interpolacji n

Na pierwszym wykresie wyraźnie widać efekt Rungego dla przypadku interpolacji Lagrange'a i węzłów równoodległych. Wraz ze zwiększeniem ilości węzłów interpolacyjnych, rośnie wartość wielomianu interpolacyjnego na krańcach przedziału, co jest powodem tak gwałtownego zwiększenia błędu. Dwie pozostałe metody pokazują się z dobrej strony i utrzymują błąd bardzo bliski zeru dla każdego badanego n.

W przypadku drugiej funkcji sytuacja jest trochę inna. Interpolacja Lagrange'a w obu przypadkach (z węzłami równoodległymi i Czebyszewa) demonstruje bardzo podobne (w zasadzie identyczne zachowanie), mianowicie obserwujemy oscylacje pomiędzy wartością bardzo bliską zera i około 65.

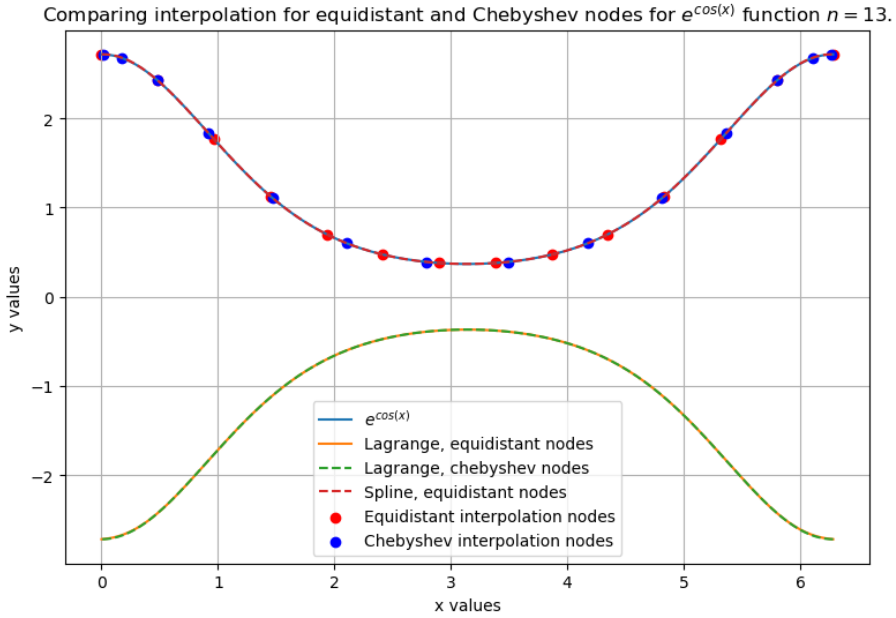
Żeby lepiej zrozumieć ten efekt, porównajmy wykres funkcji interpolacyjnych dla funkcji f_2 dla $n = 12$ (parzystego n) z wykresem dla $n = 13$ (przypadek nieparzysty).



Rysunek 4: Wykres interpolacji f_2 dla $n = 12$ - przypadek parzysty

Jak można było się domyślać, cykliczność błędu była prawdopodobnie spowodowana zmianą stopnia wielomianu z parzystego na nieparzysty. Wyraźnie widzimy, jak w przypadku parzystym wszystkie metody uzyskują bardzo dobre wyniki, a wraz ze zmianą na parzysty stopień (co za tym idzie, nieparzystą ilość węzłów interpolacyjnych) wielomiany Lagrange'a niezależnie od sposobu wyboru węzłów zachowują się tak samo źle.

Poniżej wykres dla przypadku nieparzystego. Widać złe dopasowanie.

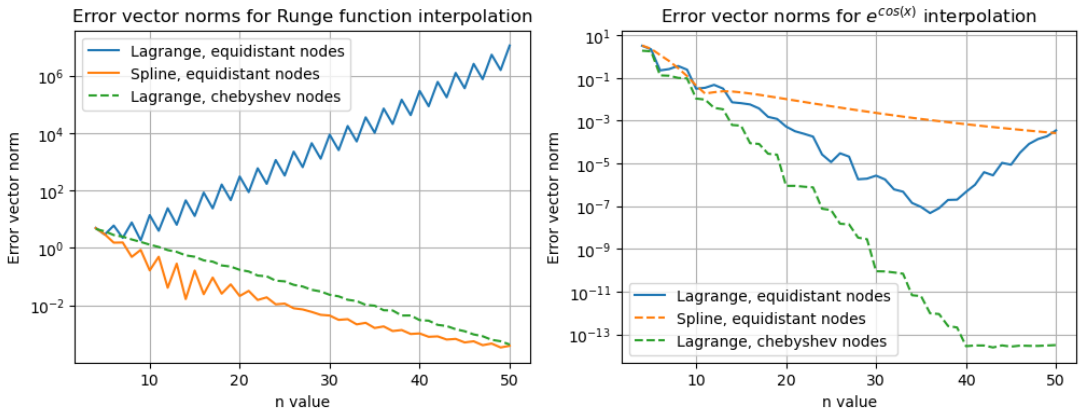


Rysunek 5: Wykres interpolacji f_2 dla $n = 13$ - przypadek nieparzysty

Nie możemy powiedzieć tego odnośnie spline'u. Spline dla wszystkich testowanych przez nas przypadków zachował precyzję niezależnie od n. Na podstawie naszych eksperymentów możemy stwierdzić, że spline jest najlepszym wyborem metody interpolacji, jeżeli bardzo zależy nam na precyzji niezależnie od funkcji interpolowanej.

Czy rzeczywiście tak to powinno wyglądać? Ponieważ naszym celem było zrozumienie zachowania badanych metod interpolacji ze względu na n i porównanie błędów tych metod, nie możemy zatrzymać się na otrzymanych wynikach, ponieważ nie są zgodne z zachowaniem, którego od nich oczekujemy (bardzo się one różnią w porównaniu do teoretycznych). Zauważyliśmy, że interpolacja Lagrange'a zachowuje się tak, jakby wykres dla przypadku nieparzystego n był odbiciem lustrzanym wykresu dla parzystej wartości n. Dlatego spróbowaliśmy wziąć wartość bezwzględną z wartości interpolacji Lagrange'a i porównać wyniki. Wyszły nam wartości bardziej sensowne. Podejrzewamy, że może to być związane z przyjętym przez nas sposobem obliczania współczynników interpolacji, który prawdopodobnie daje tutaj błędne rezultaty, aczkolwiek wydaje się być zapisany zgodnie z definicją i ciężko tu wskazać konkretny błąd.

Również przyjeliśmy skalę logarytmiczną na obu wykresach normy wektora błędów, co pozwoliło nam zaobserwować nowe zależności pomiędzy błędami badanych funkcji.



Rysunek 6: Wykres błędu w skali logarytmicznej

Na wykresie pierwszym, wyraźnie widać (tak jak i na wcześniej analizowanych wykresach), że metoda Lagrange'a z węzłami równoodległymi jest podatna na efekt Rungego.

Natomiast spline'y demonstrują najlepsze zachowanie i ich błąd cały czas maleje wraz ze wzrostem liczby węzłów. Interpolacja Lagrange'a z węzłami Czebyszewa też demonstruje dobre wyniki i wykres jej błędu jest bardzo bliski do spline'ów, maleje wraz ze wzrostem n - nie obserwujemy efektu Rungego w tym przypadku.

Na drugim wykresie błąd interpolacji Lagrange'a z węzłami równoodległymi przez pewien czas maleje (mniej więcej do $n = 36$), ale potem znów zaczyna rosnąć i dla ostatnich wartości n jego błąd jest porównywalny do błędu funkcji sklejanych, które okazały się być najgorszą interpolacją dla tego przypadku. Natomiast najlepiej pokazuje siebie metoda Lagrange'a z węzłami Czebyszewa, jej błąd cały czas maleje i tylko pod koniec dla większych wartości n zatrzymuje się na tym samym poziomie.

5 Podsumowanie i wnioski

W ramach tego laboratorium przeprowadziliśmy analizę zjawiska znanego jako efekt Rungego, które manifestuje się poprzez kontrintuicyjne pogorszenie jakości interpolacji wielomianowej, pomimo zwiększenia liczby węzłów interpolacyjnych. Naszym głównym celem było zbadanie tego zjawiska na przykładzie interpolacji dwóch funkcji opisanych we wprowadzeniu. W tym celu zastosowaliśmy trzy różne metody interpolacji: wielomiany Lagrange'a z równoodległymi węzłami, interpolację kubicznymi funkcjami sklejanymi (cubic splines), również z równoodległymi węzłami oraz interpolację wielomianami Lagrange'a z węzłami Czebyszewa.

Nasza analiza skupiła się na wyznaczeniu wielomianów interpolujących oraz spline'a dla funkcji Rungego, przyjmując odpowiednią liczbę węzłów interpolacji. Następnie dokonaliśmy próbkowania funkcji interpolowanej oraz interpolujących, a także wyznaczyliśmy wielomiany interpolacyjne. Wyniki naszych eksperymentów przedstawiliśmy na wykresach, porównując efektywność różnych metod interpolacji oraz obserwując zachowanie się błędów interpolacji w zależności od liczby węzłów interpolacyjnych.

Nasze analizy wyraźnie wykazały, że efekt Rungego jest szczególnie widoczny przy zastosowaniu interpolacji wielomianami Lagrange'a z równoodległymi węzłami. Wraz ze zwiększaniem liczby węzłów interpolacyjnych obserwowaliśmy pogorszenie jakości interpolacji na krańcach przedziału. Natomiast interpolacja za pomocą funkcji sklejanych (spline'ów) wykazała się znacznie większą dokładnością, niezależnie od liczby węzłów interpolacyjnych.

W przypadku funkcji f_1 (Rungego) obserwowaliśmy, że metoda interpolacji Lagrange'a zachowywała się podobnie zarówno dla równoodległych węzłów, jak i węzłów Czebyszewa, wykazując oscylacje w błędzie interpolacji przy nieparzystej liczbie węzłów interpolacyjnych. Natomiast spline'y zachowywały dobrą precyzję niezależnie od liczby węzłów.

Porównując błędy metod dla funkcji $f(x) = e^{cos(x)}$ w zależności od liczby węzłów, zauważyliśmy dziwną oscylację błędu w obu wariantach interpolacji Lagrange'a, która nie dawała możliwości porządnie zbadać te metody. Po narysowaniu wykresów dla n-ów parzystych i nieparzystych, zauważyliśmy dziwny błąd, kiedy w wyniku interpolacji Lagrange'a przy parzystej liczbie węzłów (n nieparzystego) dostawaliśmy funkcję interpolacyjną przemnożoną przez -1. Nie znaleźliśmy błędu w interpolacji Lagrange'a i żeby móc w jakiś sposób dokończyć nasz eksperyment, postanowiliśmy wziąć wartości bezwzględne z wartości zwracanej przez interpolację Lagrange'a. W ten sposób uzyskaliśmy wyniki podobne do wartości teorytycznych.

Na podstawie przeprowadzonych eksperymentów można stwierdzić, że interpolacja Lagrange'a z węzłami Czebyszewa była najbardziej uniwersalną metodą, pokazała siebie dobrze zarówno dla funkcji Rungego, jak i dla $f(x) = e^{cos(x)}$ dla wszystkich wartości n. Funkcje sklejane (spline'y) były dobre tylko w pierwszym przypadku, dla drugiej badanej funkcji ta metoda miała większy błąd w porównaniu do obu wariantów Lagrange'a, chociaż warto zauważyć, że błąd dla funkcji sklejanych jednak malał wraz ze wzrostem wartości n, a w interpolacji Lagrange'a z węzłami równoodległymi w pewnym momencie zaobserwowaliśmy wzrost błędu. Zarówno spline'y jak i Lagrange z węzłami Czebyszewa unikają efektu Rungego dla funkcji testowanych.

Literatura

[1] Materiały pomocnicze do laboratorium zamieszczone na platformie Teams w katalogu *lab04/lab4-intro.pdf*.

[2] Treść przedstawiona na wykładzie o interpolacji i efekcie Rungego.