MOwNiT laboratorium 1. - Sprawozdanie

Michał Szczurek
Informatya, WIEiT

Grupa poniedziałek 12:50

1 Wyznaczenie maszynowego epsilon

Zadanie polega na znalezieniu maszynowego *maszynowego epsilon*, czyli najmniejszej liczby a takiej, że $fl(1+a) > 1$. W tym celu wybrano pewną liczbę będącą potęgą 2, która po dodaniu do 1 zwróci wartość większą (w przypadku zaimplementowanej funkcji jest to 1.0). Następnie liczbę tą dzielono przez dwa, w pętli tak długo, aż suma jej i 1 była większa od 1. Wynikiem, czyli maszynowym epsilon jest obecna wartość liczby pomnożona przez 2 (jako, że maszynowy epsilon jest najmniejszą liczbą dla której suma jest większa, a nie równa 1). Poniżej znajduję się kod funkcji działającej zgodnie z opisem:

```
def find_eps(func):
    potential_eps = func(1.0)
    total = func(1.0) + func(potential_eps)

    while total > 1.0:
        potential_eps /= func(2.0)
        total = func(1.0) + potential_eps

    return potential_eps * 2
```

Jako argument przyjmowana jest reprezentacja liczby dla jakiej wyznaczony zostanie epsilon
Uzyskane w ten sposób wartości maszynowego epsilon wynoszą:

- dla domyślnej reprezentacji float **2.220446049250313e-16**
- dla np.float32 **1.1920928955078125e-07**
- dla np.float64 **2.220446049250313e-16**

Aby sprawdzić poprawność wykonanego zadania sprawdzono rzeczywistą wartość maszynowego epsilon korzystając z dokumentacji. Wartości pokrywały się z wyznaczonymi.

2 Problem ewaluacji funkcji sin(x)

Zadanie polegało na rozważeniu problemów ewaluacji sin(x) np. ze względu na zakłócenie w argumencie x.

2.1 Ocena błędu bezwzględnego przy ewaluacji sin(x)

Wartość błędu bezwzględnego, biorąc pod uwagę zakłócenie h, można wyrazić przy pomocy wzoru

$$b_{bezwzgl} = |sin(x+h) - sin(x)|$$

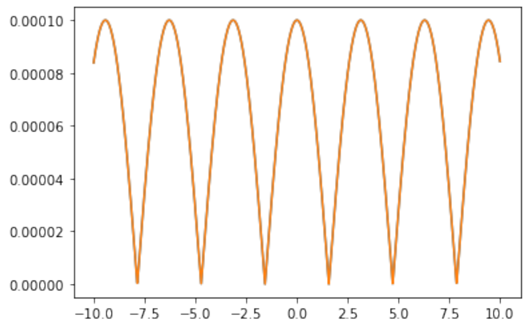
Dla odpowiednio małego zakłócenia h można zapisać

$$\lim_{h\to 0} \frac{sin(x+h) - sin(x)}{h} = sin(x)' = cos(x)$$

Wobec tego za błąd bezwzględny można przyjąć oszacowanie

$$b_{bezwzgl} = |cos(x) \cdot h|$$

Współczynnik Błąd będzie przyjmował największe wartości wtedy, gdy $|cos(x)| = 1$, czyli dla $x = k\pi$, $k \in Z$.



Rysunek 1: Wykres błędu bezwzględnego dla $h = 0.1$

Na wykresie powyżej próbowano zamieścić zarówno wartość $|sin(x) - sin(x+h)|$ jak i $|cos(x) \cdot h|$, jednak wykresy są praktycznie nierozróżnialne dla $h <= 0.1$.

2.2 Ocena błędu względnego przy ewaluacji sin(x)

Wartość błędu względnego, biorąc pod uwagę zakłócenie h, można wyrazić przy pomocy wzoru

$$b_{wzgl} = \left| \frac{sin(x+h) - sin(x)}{sin(x)} \right|$$

Wzór jest prawdziwy, dla $x \neq k\pi$, $k \in Z$.
Podążając analogicznie jak w poprzednim podpunkcie wzór można oszacować przez

$$b_{wzgl} \approx |ctg(x) \cdot h|$$

Błąd przyjmuje największe wartości dla x blisko wielokrotności $k\pi, k \in Z$, co wynika z faktu, że w tych punktach sin(x) będący w mianowniku przyjmuje najmniejsze wartości, a więc błąd jest względem niego bardzo duży (na tyle, że całkowicie obniża czytelność wykresów).

2.3 Uwarunkowanie problemu

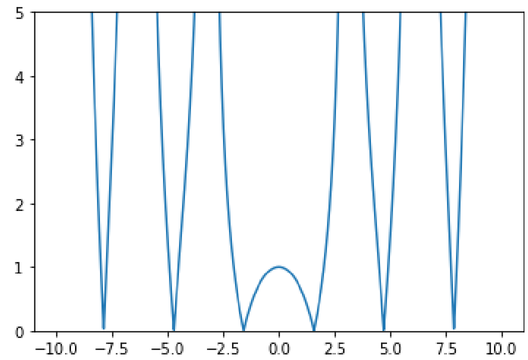
Współczynnik uwarunkowania można przybliżyć dla wzorem:

$$cond(f(x)) \approx \left| \frac{x \cdot f'(x)}{f(x)} \right|$$

Wzór ten po podstawieniu $f(x) = \sin(x)$ przyjmuje postać

$$cond(f(x)) \approx |x \cdot ctg(x)|$$

Współczynnik okresowo drastycznie zwiększa swoją wartość (czułość zadania zwiększa się) dla
$x = k\pi, k \in Z$, gdyż $\lim_{x \to k\pi} |ctg(x) \cdot x| = \infty$ (Za wyjątkiem, gdy k = 0, wtedy wartość funkcji wynosi 0). Oprócz zmian współczynnika wynikających z okresowości ctg(x) współczynnik zwiększa się również wraz ze wzrostem wartości bezwzględnej x ze względu na człon x we wzorze.



Rysunek 2: Wykres cond(f(x))

3 Przybliżanie funkcji sinus szeregiem Taylora

Celem zadania jest zbadanie błędów występujących podczas przybliżania funkcji sin(x) szeregiem Taylora wynikających z konieczności wzięcia pod uwagę tylko skończonej liczby wyrazów. W dalszej części zastosowano poniższe oznaczenia:

- y - wartość dokładna
- $\hat{y}$ - wartość przybliżona szeregiem Taylora
- Δy - błąd progresywny obliczany ze wzory $\Delta y = |y - \hat{y}|$
- Δx - błąd wsteczny obliczony przy pomocy wzoru $\Delta x = |\arcsin(\hat{y}) - x|$ korzystając z faktu, że arcsin jest funkcją odwrotną sin.

Za wartości dokładne funkcji sin i arcsin przyjęto te dostarczone przez bibliotekę numpy.

3.1 Uwzględnienie tylko pierwszego członu rozwinięcia

Przyjmując $sin(x) \approx x$ otrzymamy następujące wyniki i błędy:

- dla x = 0.1
  $y = 0.09983341664682815$
  $\hat{y} = 0.1$
  $\Delta y = 0.0001665833531718508$
  $\Delta x = 0.00016742116155979425$

- dla x = 0.5
  $y = 0.479425538604203$
  $\hat{y} = 0.5$
  $\Delta y = 0.020574461395796995$
  $\Delta x = 0.023598775598298927$

- dla x = 1.0
  $y = 0.8414709848078965$
  $\hat{y} = 1.0$
  $\Delta y = 0.1585290151921035$
  $\Delta x = 0.5707963267948966$

3.2 Uwzględnienie dwóch pierwszych członów rozwinięcia rozwinięcia

Przyjmując $sin(x) \approx x - \frac{x^3}{3!}$ otrzymamy następujące wyniki i błędy:

- dla x = 0.1
  $y = 0.09983341664682815$
  $\hat{y} = 0.09983333333333334$
  $\Delta y = 8.331349481138783e - 08$
  $\Delta x = 8.373180472587283e - 08$

- dla x = 0.5
  $y = 0.479425538604203$
  $\hat{y} = 0.4791666666666667$
  $\Delta y = 0.0002588719375363202$
  $\Delta x = 0.0002949592406357171$

- dla x = 1.0
  $y = 0.8414709848078965$
  $\hat{y} = 0.8333333333333334$
  $\Delta y = 0.008137651474563135$
  $\Delta x = 0.01488921666225429$

3.3 Wnioski

- W obu przypadkach oba błędy były największe dla $x = 1.0$ i najmniejsze dla $x = 0.1$
- W każdym przypadku błąd wsteczny był większy od odpowiadającego błędu progresywnego
- uwzględnienie dodatkowego członu szeregu miało znaczący wpływ, na zmniejszenie błędów, co widać szczególnie dobrze na przykładzie błędu wstecznego dla $x = 1.0$ - Błąd zmalał prawie czterdziestokrotnie.

4 Przykład algorytmu niestabilnego numerycznie

Przykładowym problemem, dla którego można wskazać zarówno algorytm numerycznie stabilny jak i niestabilny jest policzenie $y_n = \int_0^1 \frac{x^n}{x+10}\,dx$ dla n = 1..N. Można zauważyć, że $y_n + 10y_{n-1} = \int_0^1 \frac{x^n+10x^{n-1}}{x+10}\,dx = \int_0^1 x^{n-1}\,dx = \frac{1}{n}$. Pozwala to na wyznaczenie wzoru:

$$y_n = \frac{1}{n} - 10y_{n-1}$$

4.1 Algorytm niestabilny

Krokiem pierwszym algorytmu będzie policzenie wartości y_0.

$$y_0 = \int_0^1 \frac{x}{x+10}\,dx = ln(11) - ln(10) \approx 0.0953101798$$

Następnie na podstawie tej wartości i wzoru rekurencyjnego można policzyć kolejne N liczb. Okazuje się jednak, że dla większych n wynik znacząco odbiega od oczekiwanego. Przykładowo dla $n = 18$ wynikiem jest -91.69401033, co zdecydowanie nie pokrywa się z oczekiwaniami - liczba jest ujemna i posiada dużą wartość bezwzględną, co nie jest możliwe dla zadanej funkcji. Wynika to z faktu, że błąd jakim obarczone jest y_n jest mnożony przez -10 przy obliczaniu y_{n+1}. Tak więc stosunkowo mały błąd wykonany przy obliczeniu y_0 ma duży wpływ na obliczenie następnych wartości.

4.1.1 Algorytm stabilny numerycznie

Podstawową obserwacją, na której bazuje algorytm stabilny jest fakt, że wraz ze wzrostem n wartość całki maleje, zawsze będąc przy tym dodatnią. Różnice w wartościach dla sąsiednich n nie są znaczne (dla dużych n). Wobec tego y_{N+1} można przybliżyć przez y_N. Przekształcając wzór rekurencyjny otrzymamy

$$y_{n-1} = \frac{1}{10n} - \frac{1}{10}y_n$$

Korzystając z tego wzoru i przybliżenia można przyjąć wzór na y_N:

$$y_N = \frac{1}{10(N+1)} - \frac{1}{10} y_N = \frac{1}{11(N+1)}$$

Pozwala to przybliżyć $y_{18} \approx 0.0047846$, co jest znacznie lepszym przybliżeniem. (wartość otrzymana przy pomocy zewnętrznej strony internetowej to 0.0048066. Następnie możliwe jest obliczenie pozostałych wartości korzystając z przekształconego wzoru i y_N. Teraz podczas obliczenia kolejnych wartości (dla mniejszych n) błąd będzie dzielony przez -10, a nie mnożony. Otrzymana w ten sposób wartość y_0 pokrywa się z tą uzyskaną poprzednim algorytmem dla 15 miejsc znaczących.