

Lista Nr 04, laboratorium

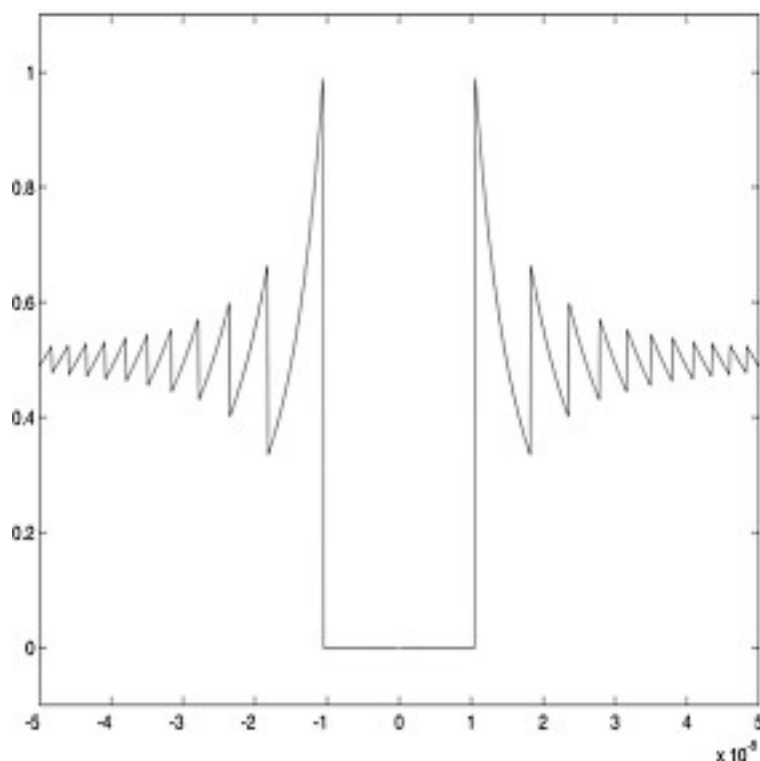
(wyk. 4 XI 10 r.)

Reprezentacja liczb rzeczywistych (*floating point*) w komputerze.

Wiadomo, że

$$\lim_{x \rightarrow 0} \frac{1 - \cos x}{x^2} = \frac{1}{2}.$$

natomiast wykres funkcji $y = \frac{1 - \cos x}{x^2}$ sporządzony przy pomocy pakietu do obliczeń *MatLab* na przedziale $[-0.5 \times 10^{-7}, 0.5 \times 10^7]$ (punkty są odkładane z krokiem $h = 10^{-10}$) wygląda następująco:



Rys. 1. Wykres $y = \frac{1 - \cos x}{x^2}$.

Na podstawie wykresu z Rys.1 granica ta powinna wynosić 0! Zaistaniała różnica wynika z przedstawienia liczb rzeczywistych w komputerze. Zadania są modyfikacją podobnych ze strony WWW wykładu¹ „*Introduction to Computer Science*”, Department of Computer Science, Princeton University, USA.

Skrypt *MatLab*owski przy pomocy którego został sporządzony wykres z Rys.1:

```
1 x = -0.5e-7 : 1e-10 : 0.5e-7;
2 y = (1-cos(x))./(x.*x);
3 axis([-0.5e-7 0.5e-7 -0.1 1.1]);
4 plot(x,y);
```

Zadania z tej listy laboratoryjnej mają na celu uczulenie na błędy jakie pojawiają się przy wykonywaniu obliczeń na komputerze.

¹ <http://www.cs.princeton.edu/introcs/home/>

1. Zadania^{2 3} o których powinno się pamiętać przy czytaniu wyniku obliczeń z monitora komputera.

(a) Po wykonaniu poniższego kodu co zostanie wyświetlone na ekranie?

```
1  double a,b;
2
3  a = 12345.0;
4  b = 1e-16;
5  if( a+b==a )
6  {
7      puts("TAK");
8  }
9  else
10 {
11     puts("NIE");
12 }
```

(b) Ile razy zostanie wykonana każda z pętli `for`:

```
1  double d;
2  int i;
3
4  puts(" dla d=0.1:");
5  i = 0;
6  for(d=0.1 ; d<=0.5 ; d = d+0.1)
7  {
8      i = i+1;
9      printf("%2i : %6.2lf\n", i, d);
10 }
11 puts("\n dla d=1.1:");
12 i = 0;
13 for(d=1.1 ; d<=1.5 ; d = d+0.1)
14 {
15     i = i+1;
16     printf("%2i : %6.2lf\n", i, d);
17 }
```

(c) Jakie liczby zostaną wyświetlone na ekranie a jakie powinny, gdyby obliczenia były wykonywane dokładnie.

```
1  int i,n,N;
2  double x;
3
4  N = 25;
5  for(i=0; i<N ; i=i+1)
6  {
7      n = 0;
8      for(x=0.0 ; x<i ; x = x+0.1)
9      {
10         n = n+1;
11     }
12     if(n != 10*i)
13     {
14         printf("%i \n", i);
15     }
16 }
```

² Patrz: <http://www.cs.princeton.edu/introcs/91float/>

³ Źródła są w archiwum L04.zip.

(d) Czy aby na pewno $\sqrt{2} \times \sqrt{2} = 2$:

```

1  if( sqrt(2.0)*sqrt(2.0)==2.0 )
2  {
3      puts("TAK");
4  }
5  else
6  {
7      puts("NIE");
8  }
```

(e) Może lepiej jest z dzieleniem:

```

1  float f;
2
3  f = (float)(3.0/7.0);
4  if( f==3.0/7.0 )
5  {
6      puts("TAK");
7  }
8  else
9  {
10     puts("NIE");
11 }
```

(f) Czy w zmiennych x i y są zapisane te same liczby:

```

1  long int x;
2  float y;
3
4  x = 16777217;          // 2^24 + 1
5  y = 16777217;
6  printf("x: %10li \ny: %12.1f \n", x, y);
```

2. *Formuła Sumacyjna Kahana* [1, tw. 8] Wszystkie elementy tablicy float X[10000] są równe liczbie 0.0001. W programie jest obliczana ich suma, która powinna wynosić 1.0.

```

1  #include <stdio.h>
2  /*-----*/
3  const int N=10000;
4  /*-----*/
5
6  // Formuła Sumacyjna Kahana
7  float KSF(int n, float X[])
8  {
9      float C,S,T,Y;
10     int i;
11
12     S = X[0];
13     C = 0.0;
14     for(i=1 ; i<n ; i=i+1)
15     {
16         Y = X[i]-C;
17         T = S+Y;
18         C = (T-S)-Y;
19         S = T;
20     }
21     return S;
22 }
23 /*-----*/
```

```

24
25 float Sum(int n, float X[])
26 {
27     float S;
28     int i;
29
30     S = X[0];
31     for(i=1 ; i<n ; i=i+1)
32     {
33         S = S+X[i];
34     }
35     return S;
36 }
37 /*-----*/
38
39 int main()
40 {
41     float S1,S2,X[N];
42     int i;
43
44     for(i=0 ; i<N ; i=i+1)
45     {
46         X[i] = 0.0001;
47     }
48     S1 = Sum(N,X);
49     S2 = KSF(N,X);
50     printf("S1-1.0: %e\n", S1-1.0);
51     printf("S2-1.0: %e\n", S2-1.0);
52     return 0;
53 }
54 /*-----*/

```

Polecenia:

- (a) Sprawdzić, która z funkcji `Sum()`, `KSF()` lepiej oblicza tą sumę?
 - (b) Wykazać, przy założeniu dokładnej realizacji działań arytmetycznych, że funkcja `KSF()` faktycznie wylicza sumę elementów zadanej tablicy.
3. Napisać program, który w zadanym ciągu znaków zamieni duże litery na małe oraz małe na duże a cyfry zastąpi znakiem ' _ '.
 4. Na bazie programu z wykładu, napisać program obliczający

$$e^A = \sum_{i=0}^{\infty} \frac{1}{i!} A^i, \quad \text{gdzie } A \in \mathbb{R}^{n \times n}, \quad n = 1, 2, 3, 4.$$

Jak należy zastąpić ∞ ?

Literatura

- [1] Goldberg, D., *What Every Computer Scientist Should Know About Floating-Point Arithmetic*, Computing Surveys, March, 1991

Wrocław, dnia 25/11/2010