Lista Nr 05, laboratorium

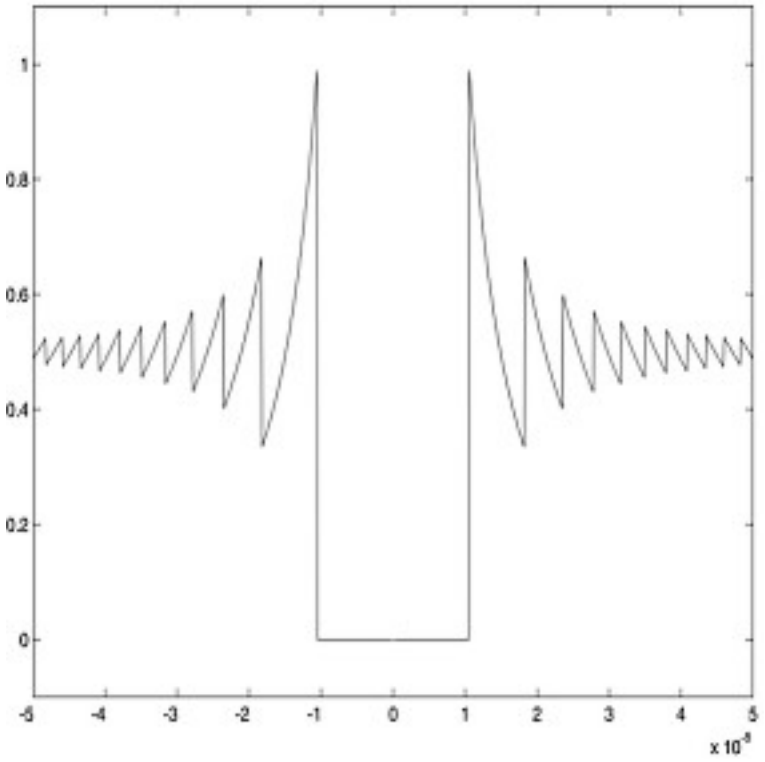
(wyk. 5,6.XI.08r)

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

Wiadomo, że

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

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 wykładu[1] *„Introduction to Computer Science"*, Department of Computer Science, Princeton University, USA.

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

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

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] o których powinno się pamiętać przy czytaniu wyniku obliczeń z monitora komputera.

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

```
double a,b;

a = 12345.0;
b = 1e-16;
if( a+b==a )
{
    puts("TAK");
}
else
{
    puts("NIE");
}
```

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

```
double d;
int i;

puts("dla d=0.1:");
i = 0;
for(d=0.1 ; d<=0.5 ; d = d+0.1)
{
    i = i+1;
    printf("%2i: %6.2lf\n",i,d);
}
puts("\ndla d=1.1:");
i = 0;
for(d=1.1 ; d<=1.5 ; d = d+0.1)
{
    i = i+1;
    printf("%2i: %6.2lf\n",i,d);
}
```

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

```
int i,n,N;
double x;

N = 25;
for(i=0; i<N ; i=i+1)
{
    n = 0;
    for(x=0.0 ; x<i; x = x+0.1)
    {
        n = n+1;
    }
    if(n != 10*i)
    {
        printf("%i ",i);
    }
}
```

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

[2] Patrz: `http://www.cs.princeton.edu/introcs/91float/`

```
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  // Formula 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[])
```

```
{
    float S;
    int i;

    S = X[0];
    for(i=1 ; i<n ; i=i+1)
    {
        S = S+X[i];
    }
    return S;
}
/*-------------------------------------------*/

int main()
{
    float S1,S2,X[N];
    int i;

    for(i=0 ; i<N ; i=i+1)
    {
        X[i] = 0.0001;
    }
    S1 = Sum(N,X);
    S2 = KSF(N,X);
    printf("S1-1.0: %+12.5e\n",S1-1.0);
    printf("S2-1.0: %+12.5e\n",S2-1.0);
    return 0;
}
/*-------------------------------------------*/
```

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

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

Jak należy zastąpić ∞?

Literatura

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

Wrocław, dnia 20/11/2008