Ćwiczenie 3 – rozwijanie w szereg i badanie zbieżności

Treść zajęć
pętla iteracyjna, stop bezwarunkowy („gwarantowany”, pętla *for* z ograniczona liczba iteracji) i warunkowy (warunkowy *break* w pętli), użycie nazw stałych matematycznych (*pi*), funkcje matlaba: *signum*, *factorial*, *mod*, *logical „and”*, uważna kontrola kopiowanego bloku (typowa pułapka dla nieuważnych),

Cel zajęć
zapis pętli iteracyjnej, warunki stopu bezwarunkowe i warunkowe, rozwiniecie w szereg Taylora funkcji „łatwo różniczkowalnej”

Wzory, algorytm:
rozwinięcie w szereg Taylora i reszta szeregu:

$$f(a+h) = f(a) + \sum_{i=1}^{n} \frac{h^i}{i!} f^{(i)}(a), \quad R_n = \frac{h^{n+1}}{(n+1)!} f^{(n+1)}(a+\beta h), \quad (0 < \beta < 1)$$

Problem:
Obliczyć wartość funkcji sin(3.2) , z błędem względnym $|\varepsilon| \leq 1 \cdot 10^{-4}$, rozwijając funkcję w szereg Taylora wokół $a = \pi$ oraz wokół 2π. Ile wyrazów szeregu należy uwzględnić w obu przypadkach? Który z szeregów jest szybciej zbieżny i dlaczego? Zamiast podanego oszacowania reszty można posłużyć się przybliżonym oszacowaniem błędu korzystając z wartości kolejnych iteracji.

Rozwiązanie:
(poniżej kod programu w matlabie)

```
% Adam Zaborski ćw. nr 3 - rozwinięcie w szereg Taylora
clc
clear all
format compact, format long

eps = 1e-6;

% rozwinięcie wokół pi
h_1 = 3.2 - pi;
sin_1 = sin(pi);    % na początek...
for i = 1: 1: 100;
   a1 = mod( i, 4 );
   if a1 == 1
      znak = -1;       % 1. pochodna itd.
   elseif a1 == 2
      znak = 0;
   elseif a1 == 3
      znak = 1;
   else
      znak = 0;
   end
   d_sin_1 = h_1^i / factorial(i) * znak;
   sin_1 = sin_1 + d_sin_1;
```

```
    if ( ( abs(d_sin_1) < eps ) & ( znak ~= 0 ) ) break;
    end;
end;
% wydruk końcowych wyników
disp('rozwinięcie funkcji wokół pi:')
disp('   liczba iteracji'), disp(i)
disp('   obliczona wartość funkcji:'), disp(sin_1)
disp('   ścisłe rozwiązanie:'), disp(sin(3.2))
disp('   błąd bezwzględny:'), disp(abs(sin(3.2)-sin_1))

% rozwinięcie wokół 2*pi
h_2 = 3.2 - 2 * pi;
sin_2 = sin( 2 * pi );
for i = 1: 1: 100;
    a1 = mod( i, 4 );
    if a1 == 1
        znak = 1;
    elseif a1 == 2
        znak = 0;
    elseif a1 == 3
        znak = -1;
    else
        znak = 0;
    end
    d_sin_2 = h_2^i / factorial(i) * znak;
    sin_2 = sin_2 + d_sin_2;
    if ( ( abs(d_sin_2) < eps ) & ( znak ~= 0 ) ) break;
    end;
end;
% wydruk końcowych wyników
disp('rozwinięcie funkcji wokół 2*pi:')
disp('   liczba iteracji'), disp(i)
disp('   obliczona wartość funkcji:'), disp(sin_2)
disp('   ścisłe rozwiązanie:'), disp(sin(3.2))
disp('   błąd bezwzględny:'), disp(abs(sin(3.2)-sin_2))
```

Wyniki:

dla rozwinięcia wokół π, $h = 0.058407346$, mamy:

nr kolejny wyrazu szeregu	1	2	3
wartość	-0,05841	0	3,32086E-05
suma częściowa	-0,05841	-0,05841	-0,05837414
błąd względny	0,000569	0,000569	-9,7029E-08

(wystarczają trzy wyrazy)

dla rozwinięcia wokół $2 \cdot \pi$, $h = -3.083185307$, mamy (bez zerowych wyrazów parzystych):

nr kol.	1	3	5	7	9
wartość	-3,08319	4,884809513	-2,321757689	0,525492906	-0,069379892
suma część.	-3,08319	1,801624206	-0,520133483	0,005359423	-0,064020469
błąd wzgl.	51,81765	-31,8633943	7,910340309	-1,091811586	0,096726479

nr kol.	11	13	15	17
wartość	0,005995704	-0,00037	1,65384E-05	-5,77996E-07
suma część.	-0,058024765	-0,05839	-0,058373581	-0,058374159
błąd wzgl.	-0,005985161	0,000274	-9,63248E-06	2,69105E-07

(potrzeba 17 kolejnych wyrazów)

Wniosek: znacznie szybciej zbieżny jest szereg dla mniejszego *h*.