Table of Contents

Aufgabe 1

```
dbtype('ex01.m');
run('ex01.m');
```

Aufgabe 2

```
dbtype('ex02.m');
a = 0.5;
b = 0.5;
c = 10.0;

if ex02(a,b,c)
    fprintf('Mit den Werten a = %f, b = %f, c = %f gilt das
 Assoziativgesetz nicht.\n', a, b, c);
else
    fprintf('Mit den Werten a = %f, b = %f, c = %f gilt das
 Assoziativgesetz.\n', a, b, c);
end
```

Aufgabe 3

```
dbtype('ex03.m');
run('ex03.m');
```

Aufgabe 4

```
dbtype('ex04.m');
run('ex04.m');
```

```
% formel fuer
% mantissenlaenge t = 3
% Exponenten p aus {-2,...,2}
% Basis b = 2

t = 3;
b = 2;
p = -2:2;

results = [];
```

```

    % jedes p_i aus p
    for p_i = p
        % jedes i aus allen moeglichkeiten 2^t
        for i = 0:2.^t-1
            bit_row = (dec2bin(i,t)-'0').';
            % mantisse berechnen
            mantisse = 0;
            for bit_idx = 1:t
                mantisse = mantisse + bit_row(bit_idx)*b.^(-bit_idx);
            end
            % mit exponenten verrechnen
            x = mantisse * b^(p_i);
            results = [results;x];
        end
    end

    % remove duplicates and sort
    results = unique(sort(results),'rows');
    results
    size(results,1)

```

```
results =

         0
    0.0312
    0.0625
    0.0938
    0.1250
    0.1562
    0.1875
    0.2188
    0.2500
    0.3125
    0.3750
    0.4375
    0.5000
    0.6250
    0.7500
    0.8750
    1.0000
    1.2500
    1.5000
    1.7500
    2.0000
    2.5000
    3.0000
    3.5000


ans =
```

```
function y = ex02(a,b,c)
    y = round(a + round(b + c)) ~= round(round(a + b) + c);
end
```

Mit den Werten a = 0.500000, b = 0.500000, c = 10.000000 gilt das Assoziativgesetz nicht.

```
%%% Aufgabe 3
%% a)
% Anonymous Function for 2^(-x)
epsilonFunc=@(k) 2.^(-k);

i = 0;
epsilon=epsilonFunc(i);
while 1 + epsilonFunc(i) > 1
    epsilon=epsilonFunc(i);
    i = i + 1;
end

fprintf('\nsmallest epsilon=%e\n',epsilon);
fprintf('\nk = %e\n\n',i);

smaller_epsilon = epsilon - 0.000001e-16;

if 1 + smaller_epsilon > 1
    fprintf('smaller epsilon exists: %e\n',smaller_epsilon);
else
    fprintf('smaller epsilon does not exist\n');
end

%% b)
% Helper Functions
dbtype('topExp.m');
dbtype('bottomFactorial.m');
dbtype('topFactorial.m');
dbtype('bottomAddition.m');
dbtype('topAddition.m');
fprintf('\n');

iterations = 0;


M = 2;
top = inf;
bottom = 0;

% fast top approximation
[top,exp_iter] = topExp(M,1.01);
fprintf('Iterationen exponentielle Aproximation: %d\n', exp_iter);
M = top;

```

```
% fprintf('Eingrenzung nach oben: %d\n', M);


% factorial approximation
change_factor = 10;
reduce = 1;
t = 1;
change_factor_func = @(x) 2.^(-0.1*x)*5+1;
change_factor = 10;

fact_iter = 0;

while round(change_factor,12) ~= 1
    change_factor = change_factor_func(t);
    if reduce
        [bottom, iter] = bottomFactorial(M, change_factor);
        fact_iter = fact_iter + iter;
        M = bottom;
        reduce = 0;
    else
        [top,iter] = topFactorial(M, change_factor);
        fact_iter = fact_iter + iter;
        M = top;
        reduce = 1;
    end
    t = t + 1;
end

fprintf('Iterationen Approximation mit Faktor: %d\n', fact_iter);

% approximation with addition
add_iter = 0;

if reduce
    [bottom,add_iter] = bottomAddition(M,2);
    M = bottom;
    %fprintf('Eingrenzung nach unten: %d\n', M);
else
    [top,add_iter] = topAddition(M,2);
    M = top-2;
    %fprintf('Eingrenzung nach oben: %d\n', M);
end

fprintf('Iterationen Approximation durch Addition: %d\n', add_iter);

% check result
C = M+2;
if (M+1)-M == 1 && (C+1)-M ~= 1
    total_iter = exp_iter + fact_iter + add_iter;
    fprintf('Supremum fuer M: %d\n',M);
    fprintf('Anzahl Schleifendurchlaeufe zur Approximation: %d\n', total_iter);
```

```
else
    fprintf('Fehler im Programm\n');
end
```

```
smallest epsilon=2.220446e-16

k = 5.300000e+01

smaller epsilon exists: 2.220445e-16
```

```
function [y,i] = topExp(M,k)
    y = M;
    i = 0;
    while (y+1)-y==1
        y = y.^k;
        i = i + 1;
    end
end
```

```
function [y,i] = bottomFactorial(M, k)
% Function approximate with factor against the condition (y+1)-y ~= 1
    y = M;
    i = 0;
    while (y+1)-y ~= 1
        y = y/k;
        i = i + 1;
    end
end
```

```
function [y,i] = topFactorial(M, k)
    y = M;
    i = 0;
    while (y+1)-y == 1
        y = y * k;
        i = i + 1;
    end
end
```

```
function [y,i] = bottomAddition(M,k)
    y = M;
    i = 0;
    while (y+1)-y ~= 1
        y = y - k;
        i = i + 1;
    end
end
```

```
function [y,i] = topAddition(M,k)
    y = M;
    i = 0;
    while (y+1)-y == 1
        y = y + k;
        i = i + 1;
```

```
        end

    end
```

```
Iterationen exponentielle Aproximation: 47
Iterationen Approximation mit Faktor: 685
Iterationen Approximation durch Addition: 11
Supremum fuer M: 9007199254740990
Anzahl Schleifendurchlaeufe zur Approximation: 743
```

```

%% summe konvergiert. maximale maschinegenauigkeit ist erricht wenn f(t) == f(t-1)
%% also wenn 1/(n^2) == 0
%% um das zu beschleunigen (approximation), kann als abbruchbedingung round(1/(n^2),x) == 0 genommen werden

f = @(x) 1/(x.^2);
n = 1;
% Runden an Nachkommastelle
r = 10;

sum_f = 0;

while round(f(n),r) ~= 0
    sum_f = sum_f + f(n);
    n = n + 1;
end

fprintf('Ergebnisse (Iterationen):\n');
fprintf('Eigene Implementierung (%d):\t%e\n', n, sum_f);

%% Alternitiv Matlab Style und wahrscheinlich genauer und schneller:

%% sum
num_iter = 10^6;
n = 1 : num_iter;
sum_f = sum(1./(n.^2));
fprintf('Matlab sum (%d):\t\t\t%e\n', num_iter, sum_f);

%% symsum
syms k
s_sum = symsum(1/k^2, k, 1, num_iter);

fprintf('Matlab symsum (%d):\t\t%e\n', num_iter, s_sum);


%% Echtes Ergebnis
fprintf('Echtes Ergebnis (PI^2)/6:\t\t%e\n', (pi.^2)/6);

%% naechster versuch mit abschaetzung M = 16666667
% kleine Zahlen zuerst Addieren, wegen floating point Ungenauigkeiten
```

```

M = single(8333332);
j = M;
sum_f = single(0);
while j >= 1
    sum_f = sum_f + f(j);
    j = j - 1;
end

fprintf('Ergebnis mit Abschaetzung (%d):\t%e\n', M, sum_f);
```

```
Ergebnisse (Iterationen):
Eigene Implementierung (141422): 1.644927e+00
Matlab sum (1000000):   1.644933e+00
Matlab symsum (1000000):  1.644933e+00
Echtes Ergebnis (PI^2)/6:  1.644934e+00
Ergebnis mit Abschaetzung (8333332): 1.644934e+00
```

Published with MATLAB® R2016a