24 May 2019

Jakub Tłuczek
FA2

Steffensen's method implementation for finding a root of the function

$$f(x) = \sum_{k=0}^{n} a_k cos(kx)$$

Project №2

Topic №11

1 Steffensen's method description

This method of finding the roots of any functions is very similar to the one described by Newton. However, method invented by Johan Frederik Steffensen is more efficient, since it doesn't use derivatives, but still manages to achieve quadratic convergence.

First, we take a guess and we treat it as x_0 value. Then, we compute $f(x_0)$ and save it as h which will be useful later. Next, we compute $g(x_0)$ which is computed by the formula:

$$g(x_n) = \frac{f(x_n + h) - h}{h}$$

We proceed to calculate x_1 using the general formula:

$$x_{n+1} = x_n - \frac{f(x_n)}{g(x_n)}$$

Next, we repeat the procedure (substituting x_1 for x_0) until $|g(x_n)|$ is smaller than the tolerance we provided in the beginning.

Finally, we can check, if the root we found, indeed produces the output equal to zero (with room for an error). This method is expected to be as efficient and fast as it is simple, and for calculating values of the function, we are going to use Groetzel algortihm.

2 Description of a program

Explanation of the code included in comments(starting with

```
function [res] = nasza(n, a, x)
res = 0;
for j=0:n
    res = res + a(j+1)*cos(j*x);
end
end
```

Now, let's see code behind Steffensen's method:

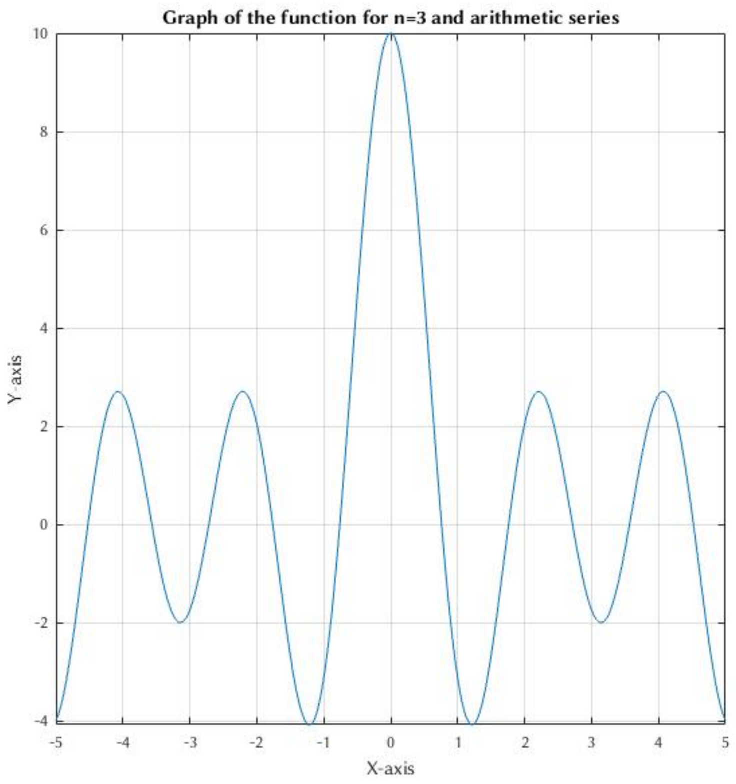
```
function [p, val] = Steffensen(n, a, p0, tol)
% Function takes following arguments:
% n - from the formula of the function
% a - series from the formula
% p0 - initial guess
% tol - tolerance
% It produces the output consisting of found root, which we can later
% substitute as an argument of our function, and array of approximations
format compact
format long
val = [p0];
for i=1:1000    %1000 is an arbitrary max, in which we expect function to
                % converge
    f = nasza(n, a, p0);
    g = (nasza(n, a, p0+f) - f)/f;
    if g == 0      %display result if g hits 0
        p
        break
    end
    p = p0 - f/g;   % calculating the next term
    val = [val, p];
    if abs(nasza(n,a,p)) < tol % display value if within tolerance
        p
        break
    end
    p0 = p;         % assign new value and proceed to next iteration
```

```
end
if abs(nasza(n,a,p)) > tol  % If failed to converge, display message
    'Failed to converge in 1000 iterations';
end
```

3 Numerical examples

Firstly, let's look at two examples of graphs of our main function:

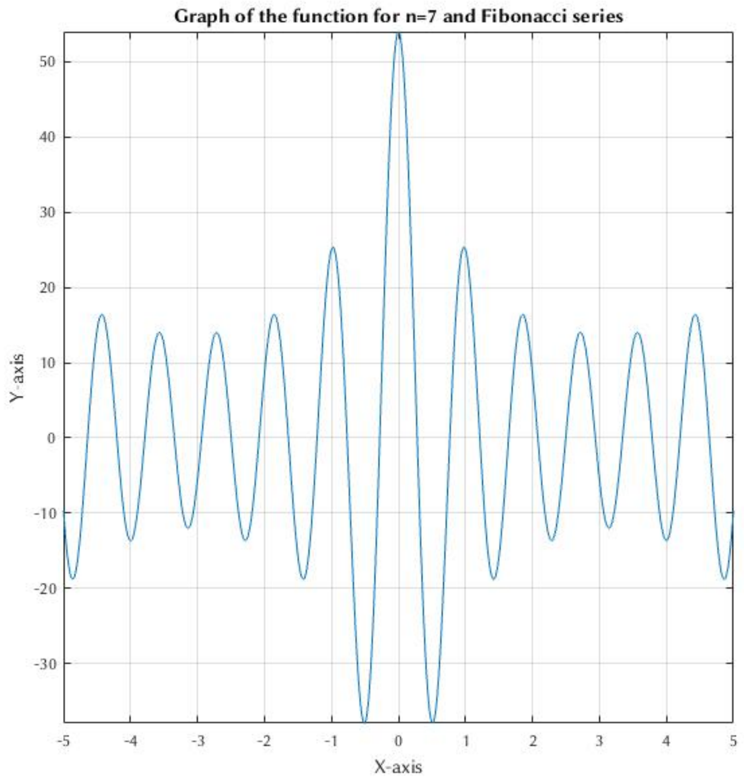
Rysunek 1: Graph 1.



Example №1 For the first example let's think about something rather simple. Let's set n to 9, and let's use arithmetical series $a_n = n$ where $i = 1, 2, 3....$ Therefore I entered following commands into MATLAB shell:

```
>> a = [1:100];
>> tic; [A, Ag] = Steffensen(9, a, 19, 0.001); toc
p =
   65.148229351506572
```

Rysunek 2: Graph 2.



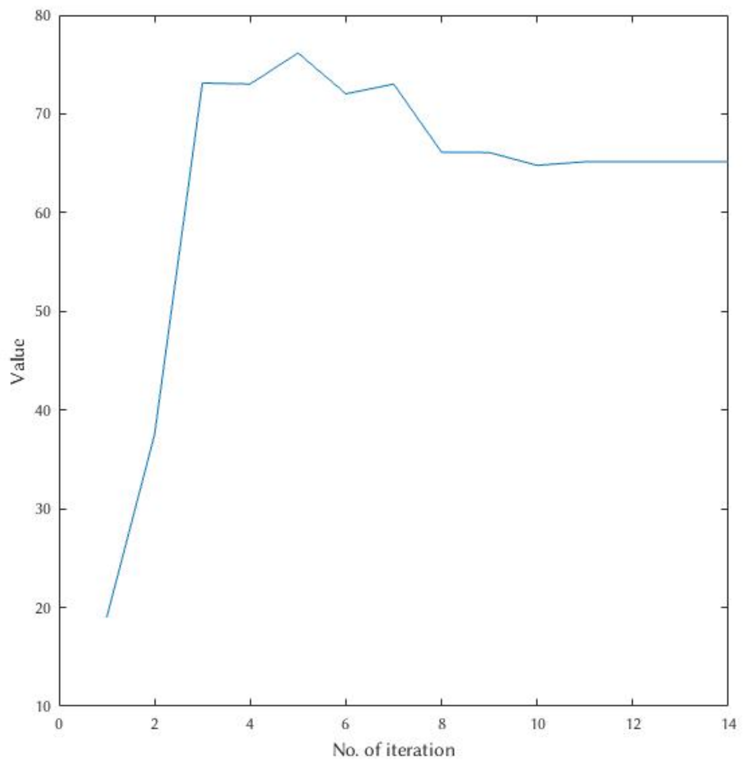
```
Elapsed time is 0.001700 seconds.
>> length(Ag)
ans =
     14
>> Guesses = [1:14];
>> plot(Guesses, Ag)
>> xlabel('No. of iteration')
>> ylabel('Value')
>> nasza(9,a,A)
ans =
      -2.216117124387296e-05
```

As we can see, solution is within our margin of tolerance, function did calulate the root pretty fast (just over 1 milisecond), and using the plot we can see how our program has been looking for the root:

Example №2 In the next example let's consider the simpler example than before, but with two different guesses to compare results.

First, let's set n to just 3, leave the arithmetical series as it was in the first example, and compute guess value 1:

Rysunek 3: Example 1.

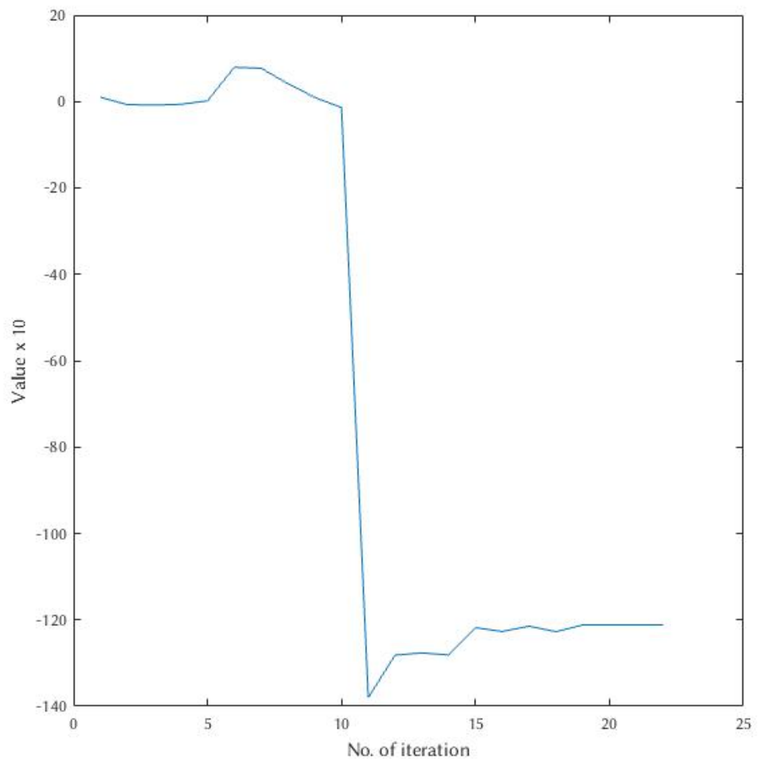


```
>> tic; [A, Ag] = Steffensen(3, a, 1, 0.0001); toc
p =
      -1.211418174377930e+02
Elapsed time is 0.002239 seconds.
>> nasza(3,a,A)
ans =
       5.159159588430384e-07
>> Guesses = [1:22];
>> plot(Guesses, Ag)
>> xlabel('No. of iteration')
>> ylabel('Value x 10')
```

Now, let's leave everything as is, but change the intitial guess:

```
>> tic; [A, Ag] = Steffensen(3, a, 5, 0.0001); toc
p =
  -57.308489754160192
Elapsed time is 0.001021 seconds.
>> Guesses = [1:41];
>> plot(Guesses, Ag)
>> xlabel('No. of iteration')
```

Rysunek 4: Example 2a.



```
>> ylabel('Value')
>> nasza(3,a,A)
ans =
       1.019283959768913e-05
```

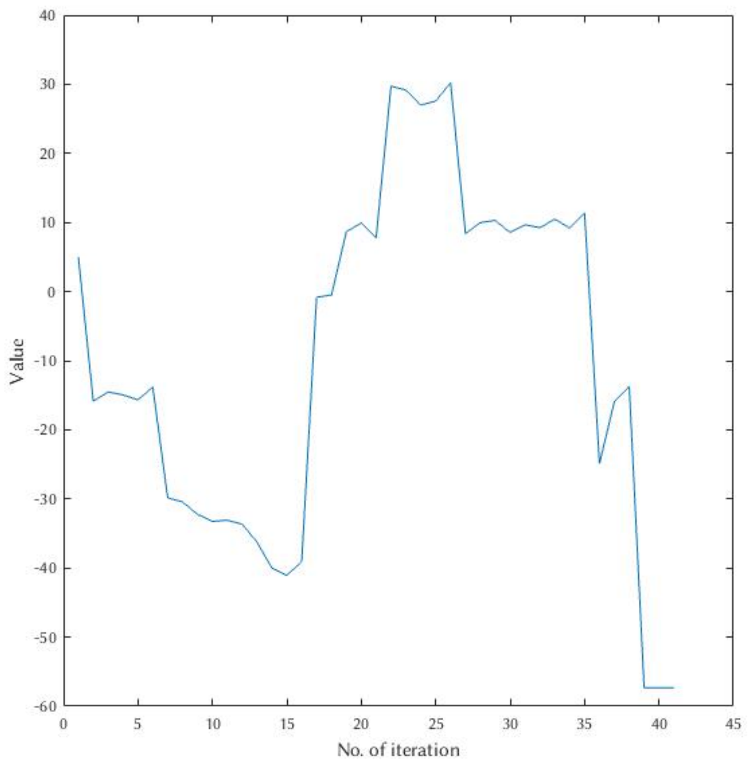
As we can see, although second guess was further away from its root our algorithm has eventually found, and it took it a greater amount of steps to compute, it still managed to do it nearly in half as much time as in the first case. We can derive two conlusions:

- Algorithm doesn't look for the closest root, but for any
- Algorithm may find root faster even with greater amount of iteration to achieve this goal

Example №3 In the third example, let's consider some example with a very big number n:

```
>> tic; [A, Ag] = Steffensen(49, a, 12, 0.0001); toc
p =
       1.732788283739346e+04
```

Rysunek 5: Example 2b.



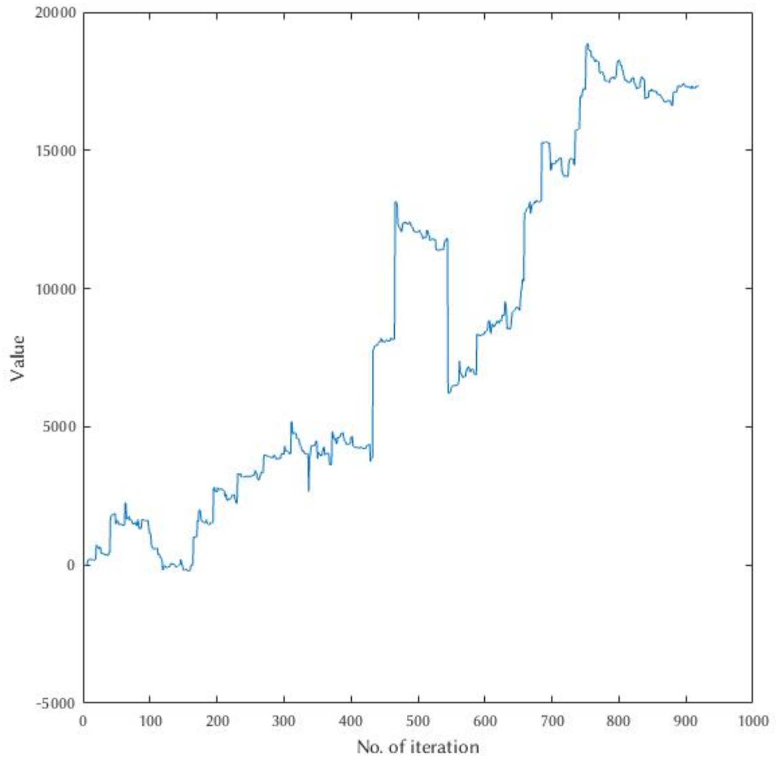
```
Elapsed time is 0.015472 seconds.
>> nasza(49,a,A)
ans =
      5.259008588609504e−07
>> Guesses = [1:919];
>> plot(Guesses, Ag)
>> xlabel('No. of iteration')
>> ylabel('Value')
```

Firstly, we can see that the time of execution is much bigger than in preceding examples. Also, the result is also significantly greater, and as we can see on the graph, the search was somewhat chaotic.

4 Analysis of the result

Overally, as we can see in the 3rd section, Steffensen's method is very quick, and can produce fast and reliable result, depending on how much tolerance we can give. Steffensen's method is often compared to the Newton's method, since they're both quadratically convergent - that is, number of correct digits in the answer doubles every iteration. As we can see, for this particular function it takes at most ca. 0.1s.

Rysunek 6: Example 3.



Unfortunately, there are some drawbacks of this method. Firstly we have to calculate both $f(x_n)$, as well as $g(x_n)$, which also depends on f. In this very example function isn't that complicated really, but with more complex ones, as well as with the bigger data sets, it'll slow down our computations significantly.

Also, as with every iterative root finding algorithms, it really depends on the choice of x_0 how fast (or if) algorithm will find a root. Sometimes it will find a value which is good enough, but isn't a root. On other ocassions it may diverge to infinity, which is not satisfactory at all. Also, it may jump between two potential values until it will run out of time, or to be more precise - number of tries, which is specified in the argument of the main loop.

Overally though, it has to be said, that Steffensen's method is incredibly fast and accurate enough to use it in projects to speed up our calculations.

Reference

[1] L.W. Johnson, D.R. Schulz, On Steffensen's method, SIAM, 1968.

[2] Gerald Goertzel, An Algorithm for the Evaluation of Finite Trigonometric Series, The American Mathematical Monthly, January 1958.

[3] Germund Dahlquist, Åke Björck, Beräkningsvetenskap (Numerical analysis), 1974, translated by Ned Anderson.