The Weierstrass method for finding all roots of the Chebyshev polynomial of the first order

Kornel Skórka & Julian Szachowicz

May 2019

1 Task Description

The assigned task is as follows;
The Weierstrass method for finding all roots of the polynomial

$$w(x) = p_1 T_n(x) + p_2 T_{n-1}(x) + ... + p_{n+1} T_0(x)$$

where $T_k(x)$ is the Chebyshev polynomial of the first order.

2 Method Description

The Weierstrass method is an iterative method closely related to Newton's method for finding the roots of a polynomial. In a simplified manner, the method is brought down to the following expression:

$$x_i^{k+1} = x_i^k - \frac{w(x_i^k)}{\prod_{j=1, i \neq j}^{n} (x_j^k - x_i^k)}$$

It is important to realize, that we are not dealing with a single simple polynomial, but a linear combination of Chebyshev polynomials of the first order, thus the roots are actually representable in the following way:

$$x_i \approx \alpha_i = cos(\frac{2i+1}{2n}\pi) \text{ for } i = 1, 2, 3, ...n$$

Thus, the program aims to find, based on approximated roots on input, to find the precise values for which $w(x_i) = 0$.

3 Program Description

We have strived to create the most efficient and compact approach to the task and so in the end we have created three vital functions, which are described below.

- function [x,k] = Weierstrass(x0, p, tol, max_iter) - the main function, which computes the roots of a polynomial formed of Chebyshev polynomials.

The above function uses an internal function to calculate the product of differences, which is:

- function s = rootsProduct(x, i) - Helper function to calculate the product of the differences of root approximations.

We make use of an external function, which calculates simulates the function and returns the value of $w(x)$ for a given x.

- function w = WeierstrassPolynomial(x,p) - computes the value of the Weierstrass+Chebyshev polynomial for the given value of x

The general approach to the problem should be considered as:

1. User creates an array of coefficients (of size $n+1$) and an array of different initial approximations to x
2. User calls the function Weierstrass to conduct iteration on the approximations
   (a) The program iterates on the approximates, slightly changing the values in the direction of the real roots of the polynomial on each iteration
   (b) The program exits when the norm of the array of changes is smaller than the user-defined tolerance (ie. the change is within bounds of error) or when a maximum amount of iterations have been completed
3. User then uses WeierstrassPolynomial to check the accuracy of the iterations for the final set of x
   (a) The Chebyshev polynomials are computed in a recursive manner, as to save computer resources
4. (Optionally) User may plot the Weierstrass+Chebyshev polynomial

4 Examples

Together with the solution, we have prepared three example scripts to showcase the operation of the script.

In all examples we assume the first coefficient to be $\frac{1}{2^{n-1}}$, as this eliminates the constant which appears naturally in the the Chebyshev polynomial. The other coefficients are trivial, as their nature does not largely influence the Weierstrass+Chebyshev polynomial.

4.1 Example one

The first example we have prepared utilizes the following linearly combined Chebyshev polynomial (with $n = 3$):

$$w(x) = \frac{1}{4}T_3(x) + 2T_2(x) + 3T_1(x) - 1$$

The initial approximations for the roots x_i are chosen by the random function in an interval of $[0, 1]$. The initial values were:

toleration	$1e-6$
maximum number of iterations	100
values of initial root approximations	$x_i = [0.1190, 0.4984, 0.9597]$

With the results being as follow:

final root approximations	$x_f = [0.5993, -2.8308, -1.7685]$
number of iterations	16
values of function	$1.0e-13 * [0.3118, 0.0008, 0.0971]$

The values of $w(x)$ for the resulting approximations were very close to 0, if not 0 entirely. This is the graphed function, consider the vertical lines to be the final root approximations.

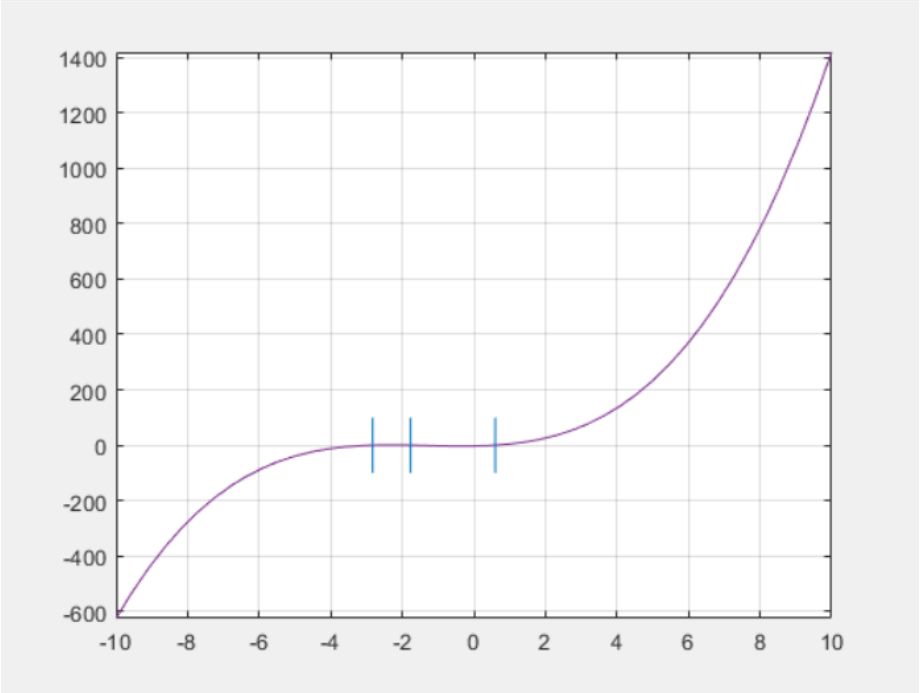


Figure 1

4.2 Example two

The second example utilizes the linearly combined Chebyshev polynomial (with (n= 7)):

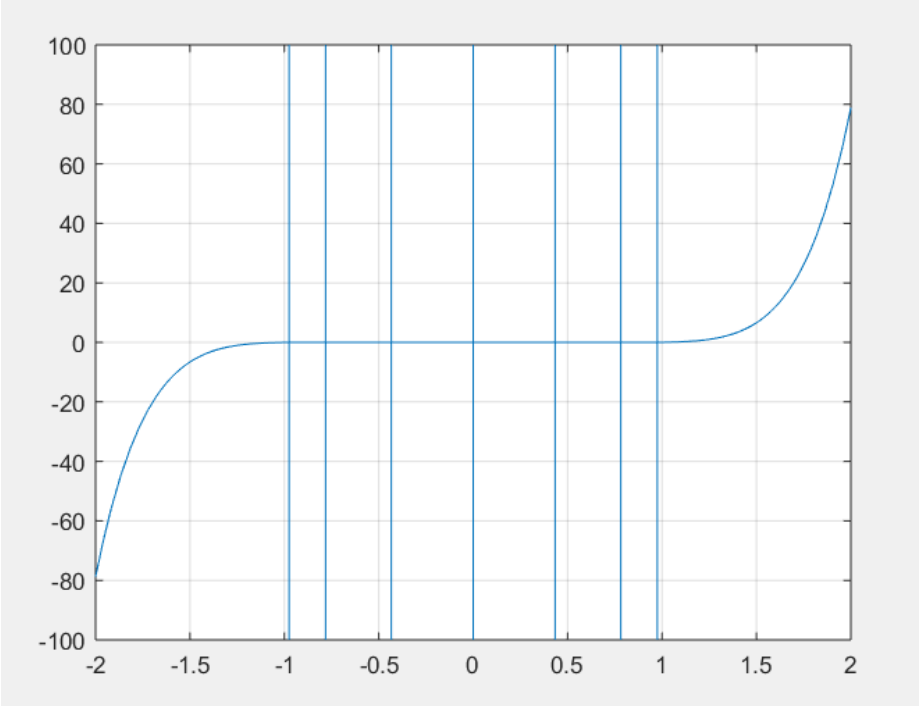
$$w(x) = \frac{1}{2^6} T_7(x)$$

The initial approximations for the roots x_i are chosen by the function $x_i(i) = cos((2 * i - 1) * \pi/(2 * n))$ The initial values were:

toleration	$1e-6$
maximum number of iterations	100
initial root approximations	$x_i = [0.9749, 0.7818, 0.4339, 0.0000, -0.4339, -0.7818, -0.9749]$

With the results being as follow:

final root approximations	$x_f = [0.9749, 0.7818, 0.4339, -0.0000, -0.4339, -0.7818, -0.9749]$
number of iterations	1
values of function	$[3.4694e-18, 0, 6.9389e-18, -5.9253e-18,$
6.9389e-18, -5.5511e-17, -7.6328e-17]	



4.3 Example three

The final example utilizes the linearly combined Chebyshev polynomial (with n=3):

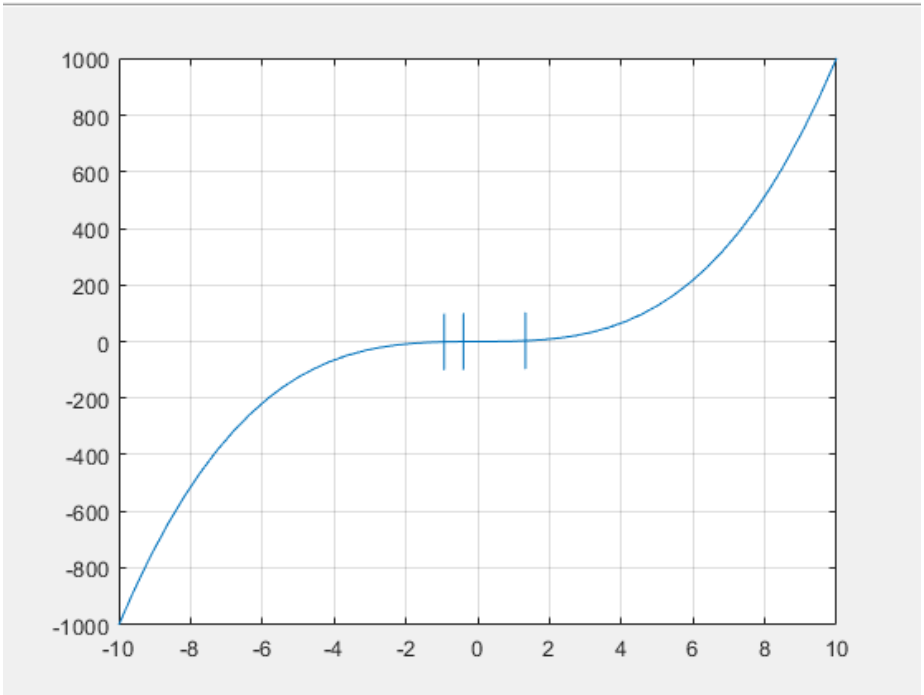
$$w(x) = \frac{1}{4}T_3(x) + T_1(x)$$

The initial approximations for the roots x_i are chosen by the function $xi = rand(n, 1) * i + randn(n, 1) * 1e - 10$ The initial values were:

toleration	$1e-6$
maximum number of iterations	100
initial root approximations	$x_i = [0.7725, 2.5222, 0.7628]$

With the results being as follow:

final root approximations	$x_f = [-0.3974, -0.9333, 1.3308]$
number of iterations	100
values of function	$[-0.1621, -1.0464, 2.6895]$



5 Analysis and conclusion

The weierstrass method for finding the roots of a polynomial-type equation (in this case a linear combination of Chebyshev polynomials) is not a satisfactory

method.

It is only in the case that the initial approximations of the roots is very close to the real roots, that this method truly works. In most cases, if the roots are too far off of the real values, the method iterations lead to divergence and very strange results.

If we look at the first example, we can notice that within 16 iterations the approximations of the roots reached a final value (one of which was within the range of tolerance). As we can see the values of the polynomial reach a very small (nearly zero) amount. In fact, since we assume this method is to find the roots of the polynomial, the absolute values of the function values is the error of the function (including the tolerance range).

If we look at the second example, we can see that we now assume root approximations to be the exact roots of the given Linear Chebyshev polynomial (which we know thanks to the nature of Chebyshev polynomials - the equation given in the introduction). The script finishes in one iteration (without actually altering the values of x), as there is no need to change the values of the approximations. The values of the function (the aforementioned error) appears to be none zero, but this is due to the method of calculating the Chebyshev polynomials. The recursive method is much more efficient and faster, but loses on accuracy. That is why the error is extremely small (almost zero).

Finally, the third example shows us that this method is not entirely reliable. After the user-given maximum iterations limit (set to 100), the method still did not reach the desired result of the exact roots for which $w(x) = 0$. Even though their values for w are small, they are still high compared to the results of the other examples. In this example the values we chose are actually complex numbers (to satisfy the complex roots of the x-form of the Chebyshev polynomials). Despite this, the values of the function for the final approximations do not reach 0.

In conclusion, the method is only viable if used for approximations very close to the real roots of the given linear Chebyshev Combination. Other methods, such as Newton's or the Halley method may prove to be more useful given the strict requirments of the Weierstrass method.

6 Appendix

Weierstrass.m

```
1 function [x,k] = Weierstrass(x0, p, tol, max_iter)
2 % [x,k] = Weierstrass(x0, p, tol, max_iter)
3 %   computes the roots of a polynomial formed
4 %   of Chebyshev polynomials.
```

```
 5 %
 6 % Output
 7 %   x :          calculated solutions
 8 %   k :          iterations completed
 9 %
10 % Input
11 %   x0 :         array of initial approximations
12 %   p :          array of coefficients
13 %   tol :        error tolerance
14 %   max_iter :   maximum number of iterations
15
16 n = max(size(p)) - 1;
17
18 dx=ones(n,1);
19 k=0;
20
21 while norm(dx)> tol && k<= max_iter
22     for i = 1:n
23     dx(i) = WeierstrassPolynomial(x0(i), p)/Product(x0,i);
24
25     end
26     x0=x0-dx;
27     k=k+1;
28 end
29
30 x = x0;
31 end
32
33 function s = rootsProduct(x, i)
34 % s = rootsProduct(x, i)
35 %   Helper function to calculate the product
36 %    of the differences of root
37 %   approximations.
38 %
39 % Output
40 %   s : product result
41 %
42 % Input
43 %   x : array of roots
44 %   i : ith root to base differences around
45
46     s = 1;
47     for j = 1:max(size(x))
48         if(i == j)
49             continue
50         end
```

```
51      s = s*(x(j)-x(i));
52      end
53  end
```

WeierstrassPolynomial.m

```
 1  function w = WeierstrassPolynomial(x,p)
 2  % w = WeierstrassPolynomial(x,p)
 3  %   computes the value of the Weierstrass+Chebyshev
 4  %   polynomial for the
 5  %   given value of x
 6  %
 7  % Output
 8  %   w : value for input x
 9  %
10  % Input
11  %   x : point for function value
12  %   p : array of coefficients of the polynomial
13
14  n = max(size(p)) - 1;
15  t = ones(n, 1);
16
17  if(n == 0)
18      w = p(1);
19      return
20  end
21
22  t(1) = x;
23  if(n == 1)
24       w = t(1) * p(1) + p(2);
25       return
26  end
27
28  t(2) = 2 * x^2 - 1;
29  if(n == 2)
30      w = t(2) * p(1) + t(1) * p(2) + p(3);
31      return
32  end
33
34  w = p(n+1) + t(1) * p(n) + t(2) * p(n-1);
35  for i = 3:n
36       t(i) = 2 * x * t(i-1) - t(i-2);
37       w = w + p(n-i+1) * t(i);
38  end
39
40  end
```

example1.m

```
n = 3;
p = [0.25, 2, 3, -1];
xi = rand(n, 1);

[xf, k] = Weierstrass(xi, p, 1e-6, 100);

fplot(@(x) WeierstrassPolynomial(x, p), [-10, 10])
grid on
for i = 1:n
    hold on
    w = WeierstrassPolynomial(xf(i), p);
    line([xf(i) xf(i)],[w - 100, w + 100]);
end
```

example2.m

```
 n = 7;
p = [1/(2^6), zeros(1, n)];
for i = 1:n
    xi(i) = cos((2*i-1)*pi/(2*n));
end

[xf, k] = Weierstrass(xi, p, 1e-6, 100);
fplot(@(x) WeierstrassPolynomial(x, p), [-2, 2])
grid on
for i = 1:n
    hold on
    w = WeierstrassPolynomial(xf(1, i), p);
    line([xf(1, i) xf(1, i)],[w - 100, w + 100]);
end
```

example3.m

```
n = 3;
p = [0.25, 0, 1, 0];
xi = rand(n, 1) * i + randn(n,1) * 1e-10;

[xf, k] = Weierstrass(xi, p, 1e-6, 100);

fplot(@(x) WeierstrassPolynomial(x, p), [-10, 10])
grid on
for i = 1:n
    hold on
    w = WeierstrassPolynomial(xf(i), p);
    line([xf(i) xf(i)],[w - 100, w + 100]);
end
```

7 Bibliography

1. Gyurhan H. Nedzhibov, "Local Convergence of the Inverse Weierstrass Method for Simultaneous Approximation of Polynomial Zeros", Faculty of Mathematics and Informatics Shumen University

2. Tateaki Sasaki, "Durand-Kerner method for the real roots", Japan Journal of Industrial and Applied Mathematics, 2002