F.D. Brunet de Rochebrune (2619225) Dynamics and Computation

Assignment 2

2019-04-17

Introduction

The root finding problem is one of the most relevant computational problems. There are several methods for root finding, the most commonly used ones are, Bisection, Secant method and Newton's method, these all converge to the root at different rates.

The Bisection method uses a function $f(x) = 0$ on a closed interval $[a, b]$, such that $f(a) \cdot f(b) < 0$, then $f(x) = 0$ has at least one root in the interval. The method repeatedly halves the interval containing the root, eventually converging to the root. This is a slow method, however it will always find the root correctly.

The Secant method constructs a straight line that intersects the curve in at least two points, also known as a secant, it then uses the root of this line to construct a new Secant between two points. Finally converging to the root of the curve.

The Newton method finds the tangent line of the function at the current point and the root of the tangent line as the next reference point, eventually converging to the root. This method can be described by: $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$. This is a fast method, however you need to know the derivative of the function.[1]
In this assignment the focus will be on the Secant method.

Syntax

For this function there are several input arguments, firstly, the function of which the root has to be calculated has to be given. Furthermore, two initial values have to be given for the Secant method to work. It is optional to define the precision of the calculation and the number of iterations.

The output is the root which has been found through calculations, as well as the number of iterations and the precision with which the root has been calculated.

Description of the Method

As explained in the syntax, the function calls for at least three and at most six input arguments. If the number of required iterations is not given, the default value is set to 10000. If the input argument tolerance is not given, this is initialized to 0.001. Furthermore, the two starting points are defined as $x(1)$ and $x(2)$. The function starts with a for loop, starting at i = 3, until the defined amount of iterations. In this for loop the Secant method formula is used,

$$x_i = x_{i-1} - f(x_{i-1}) \cdot \frac{(x_{i-1} - x_{i-2})}{f(x_{i-1}) - f(x_{i-2})}$$

[1]J.C.,, E. (2014). Comparative Study of Bisection, Newton-Raphson and Secant Methods of Root- Finding Problems. IOSR Journal of Engineering, 4(4), pp.01-07.

This for-loop continues calculating the new x_i value until the required tolerance is achieved, this is tested by calculating the absolute value of the difference between the two previous x-values. The found value of the root will be displayed, with 32 digits, by the function `vpa`. Accompanied by the number of iterations and with which precision this root has been calculated.

Furthermore, if the maximum number of iterations has been reached, without finding the root with the required tolerance, a message will be displayed saying that the Secant method stopped without converging to the desired tolerance, due to reaching the maximum number of iterations as given by the user.

Some function lines are added to allow for the possibility of showing the Secant method as an illustration.

Illustration

Applying the Secant method on the function $x^2 - 10$ with starting values $x_1 = 1$ and $x_2 = 5$ and default settings gives us the output below and Figure 1 (see next page).

```
>> secant(@(x) x.^2 - 10, 1, 5)
ans =
3.16227760800842992623814541622761
```

The MATLAB function plots the function using the `line` function, evaluating a hundred values of the input function using `linspace`. The interval of these values is chosen to be x_1 to x_2, as the root has to be between these values. This gives us a nicely zoomed-in plot. The x- and y-axis have a red line. The function is plotted in blue and the Secant lines are black.

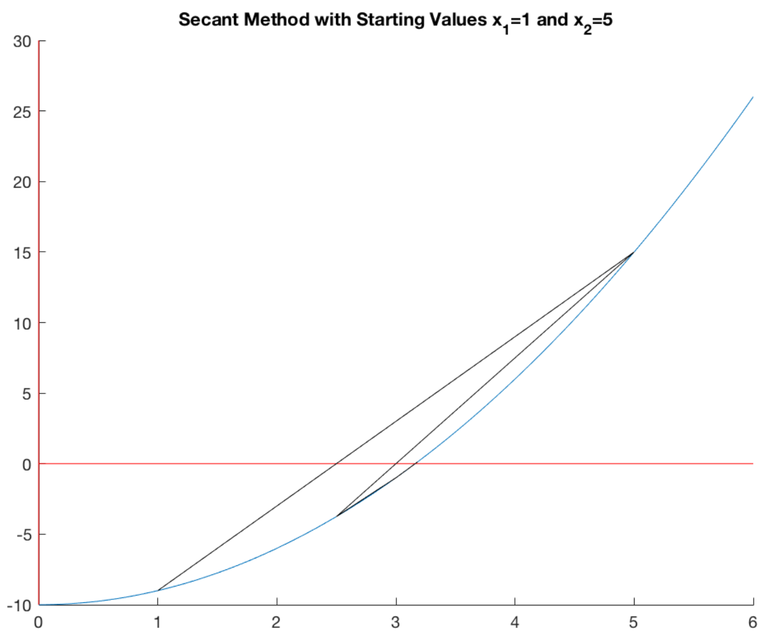

Figure 1: Graphical Representation of the Secant Method on the Function $x^2 - 10$ with Starting Values $x_1 = 1$ and $x_2 = 5$.

Questions

Order of Convergence

$x_n \to \alpha$ converges at order p if there is a C (independent of n) such that

$$\mid x_{n+1} - \alpha \mid \leq C \mid x_n - \alpha \mid^p$$

for large n. The order of convergence can be calculated by using two consecutive iterations as data, for which

$$p \approx \frac{\ln(\frac{x_{n+1}-\alpha}{x_n-\alpha})}{\ln(\frac{x_n-\alpha}{x_{n-1}-\alpha})}$$

The convergence for the bisection method is $p = 1$, for Newton's method it is $p = 2$. The order of convergence for the Secant method is $p = \frac{1+\sqrt{5}}{2} \approx 1.618$, also known as the golden ratio (φ).[2] [3]

[2]J.C.„ E. (2014). Comparative Study of Bisection, Newton-Raphson and Secant Methods of Root- Finding Problems. IOSR Journal of Engineering, 4(4), pp.01-07.

[3]van den Berg, J.B., Planqué, B., 11 January 2017, Lecture Notes on Numerical Mathematics

Advantages and Drawbacks

The main disadvantage of using Newton's method is having to calculate the derivative, the Secant method takes away this disadvantage as only $f(x)$ has to be evaluated. Newton's method requires two function evaluations per iteration, the Secant method only requires one evaluation per iteration. Therefore, this is often found to be faster, however, Newton's method often converges quicker. However, there is a drawback, as two initial values have to be given by the user. Furthermore, if the two initial values are chosen such that the Secant line has no root, i.e. the x-axis is parallel to the Secant line, the Secant method will not work.[4]

Performance

The code was written as simple and small as possible, while keeping performance in mind. The method needs a loop to calculate the secant lines. The `newton.m` method was implemented with a while-loop, so we followed. However, after a code-rewrite, an implementation with a for-loop resulted in a smaller code footprint and – after benchmarking – faster performance than a while-loop.

[4] J.C.„ E. (2014). Comparative Study of Bisection, Newton-Raphson and Secant Methods of Root- Finding Problems. IOSR Journal of Engineering, 4(4), pp.01-07.

Appendix: Function Implementation

```matlab
function [zero, niter] = secant(f, x1, x2, tol, nmax, varargin)
%SECANT Find function zeros.
%   ZERO=SECANT(FUN,X1,X2,TOL,NMAX) tries to find the zero ZERO of the
%   function FUN using the Secant method, an algorithm that uses a
%   succession of roots of secant lines as an approximation of the real root.
%   If the search fails an error message is displayed. FUN can also be an
%   inline object.
%   [ZERO,NITER]= SECANT(FUN,...) returns the iteration number at which
%   ZERO was computed.
%
%   Default values of tolerance and maximum iterations are 0.001 and 1000
%   respectively.

if nargin < 5
    nmax = 1000;
end
if nargin < 4
    tol = .001;
end

x(1) = x1;
x(2) = x2;

line(linspace(x1 - 1, x2 + 1), feval(f, linspace(x1 - 1, x2 + 1), varargin{:})); % the function
line([0 0], ylim, 'Color', 'red'); % x-axis
line(xlim, [0 0], 'Color', 'red'); % y-axis
title(['Secant Method with Starting Values x_1=' num2str(x1) ' and x_2=' num2str(x2)]);
line([x(1), x(2)], [feval(f, x(1), varargin{:}), feval(f, x(2), varargin{:})], 'Color', 'black');

for i = 3:nmax
    x(i) = x(i - 1) - (feval(f, x(i - 1), varargin{:})) * ((x(i - 1) - x(i - 2)) / (feval(f, x(i - 1), varargin{:}) - feval(f, x(i - 2), varargin{:})));
    line([x(i - 1), x(i)], [feval(f, x(i - 1), varargin{:}), feval(f, x(i), varargin{:})], 'Color', 'black');
    if abs(x(i) - x(i - 1)) < tol
        niter = i - 2;
        zero = vpa(x(i));
        return
    end
end

fprintf(['secant stopped without converging to the desired tolerance', ...
  'because the maximum number of iterations was reached\n']);
return
```