UPC - BARCELONA TECH
Master on Numerical Methods in Engineering
Fall 2016

Numerical Methods for PDEs

HOMEWORK # 1: BASICS

Jaume Betran
Samuel Parada Bustelo
Magdalena Pérez Lanfranco

Due date: November, 28th

Problem 1

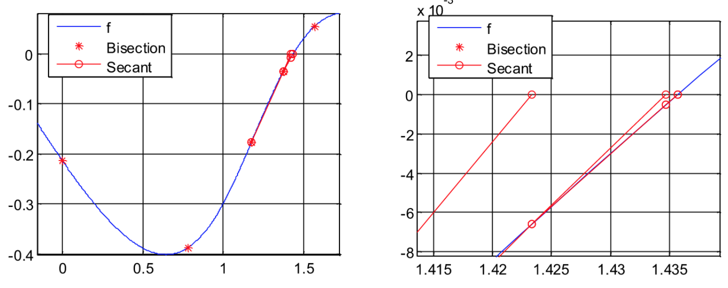
A Bisection-Secant method has been implemented that swaps from one method to the other in both directions with the following swap criteria:

- Given the initial range where the solution is continuous and a unique solution is guaranteed, one Bisection loop is performed yielding a nested range and two abscissa points to run a Secant loop. The method is unconditionally swapped to run a Secant loop. This guarantees for smooth functions that the method quickly starts a quadratic convergence.
- Using the two abscissa points provided by the last Bisection loop, a succession of Secant loops is performed yielding a succession of new approximations to the solution. In case the value of the newest approximation lays out of the original range provided by the immediately precedent Bisection loop, then a further Bisection loop is forced starting from the range yielded by the last Bisection loop. Otherwise the succession of Secant loops continues until a precision criterion is reached. This prevents the method from looking for the solution out of the range delimited by the last Bisection loop.

A further swap to a Bisection loop could be implemented to enhance the performance of this solver. In the latter succession of Secant loops the error, estimated as the distance between the two last solutions is normally decreasing. In case the newest solution lays in the range delimited by the last Bisection loop but the error does not reach the threshold nor observes a quadratic decrease, a new Bisection loop could be forced. This new Bisection loop could start from the last range defined or it could take advantage of the successive approximations yielded by the Secant loop. This part of the algorithm further guarantees that the process is not stuck locally for tricky functions.

This team originally implemented a Newton method with a numerical evaluation of the local derivative. For continuous function with monotonic derivative this method has quadratic convergence. However, if its derivative has an extreme near the zero of the function, the convergence is not guaranteed. The present Bisection-Secant method overcomes this type of difficulties with the swap criteria explained above. That guarantees sufficient number of nested Bisection loops to bound the solution in a smooth enough range.

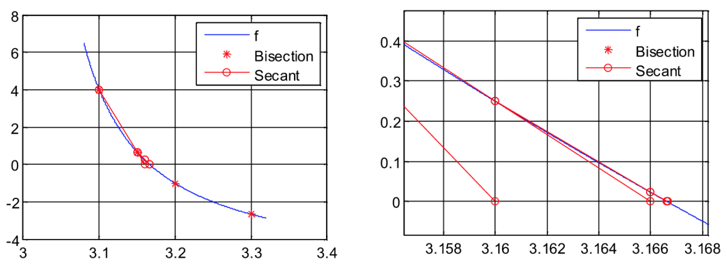
However, both methods require the solution to be unique in the local range and are not robust if multiple solutions are present. Two different convergence plots are shown in Figure 2 and Figure 3 with two different initial ranges.



(a) (b)

Figure 1: (a) Convergence scheme of the Bisection-Secant method applied to the pool problem starting from a range $[0, \pi/2]$. (b) Zoom in the third Secant loop

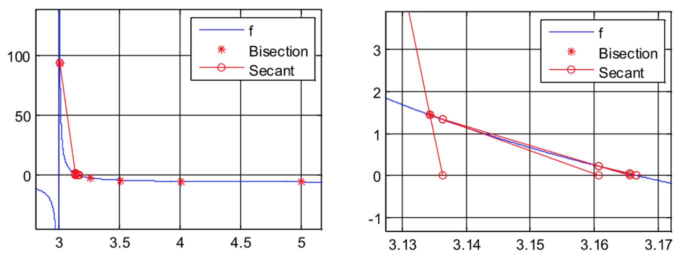
.



(a) (b)

Figure 2: (a)Convergence scheme of the Bisection-Secant method applied to the double asymptotic function starting from a range $[3.1, 3.3]$. (b) Zoom in the third Secant loop.

.



(a) (b)

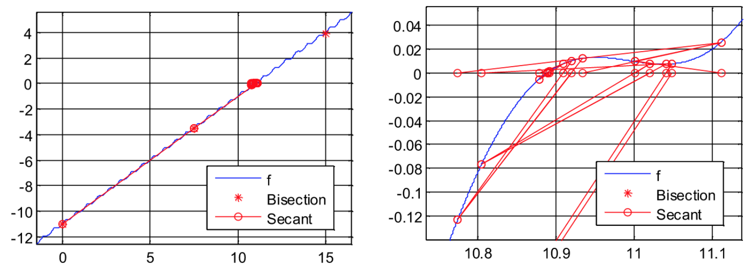
Figure 3: (a) Convergence scheme of the Bisection-Secant method applied to the double asymptotic function starting from a range $[3.01, 5.0]$. (b) Zoom in the second Secant loop.

.

Other intentionally tricky functions have been explored to show the need of some further means to bypass slow convergence issues. For instance, the family of functions

$$y = x + (0.1 + \epsilon) \cdot \sin(10x) - \frac{7}{2}\pi + \epsilon$$

has been explored for several different values of ϵ . The lower ϵ, the more complex and slow is the convergence. This function features many inflection points near the zero and this leads the solver to fail near the solution. The proposed swap to Bisection loop would probably solve the slow convergence of this specific function.



(a) (b)

Figure 4: (a) Convergence scheme of the Bisection-Secant method applied to y with $\epsilon = 0.01$ starting from a range $[0, 5\pi]$. (b) Zoom in the slow convergence region of Secant loops

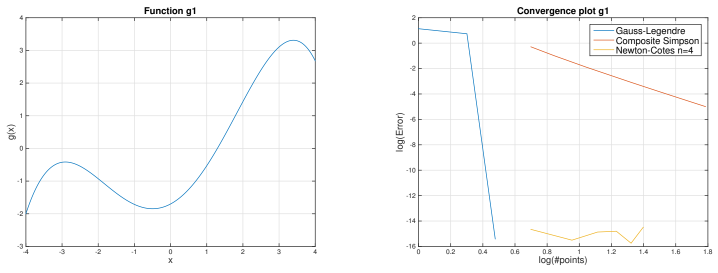
.

Problem 2

Integral 1 (Integral_1.m)

Upon first glance to the plot of the function, it looks like a polynomial of order 4 or higher, (see Figure 5). Two approaches were tested for the solution, as it is provided in the script.

First, a Newton-Cotes [1] quadrature is chosen. Composite Simpson's rule can be used to approximate the integral, but the required number of intervals needs to be incremented to get a valuable result. Same happens when using second Simpson's rule but, when a Newton-Cotes quadrature with $n = 4$ is chosen, the error becomes really small ($E = 2.2065e-15$). In addition, a Gauss quadrature [2] is also used. Effectively, when 3 points are used to evaluate the function, the error lays at machine error level. As long as the smallest amount of points is to be used, the Gauss quadrature is the preferred method.



(a) Function plot between -4 and 4 (b) Convergence

Figure 5: Plots for function $g1(x)$

Integral 2 (Integral_2.m)

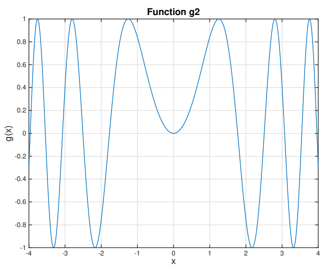
Function $g2(x)$ is symmetric. Thus, computing just half of the integral is a reasonable approach to save computational cost. Composite Simpson's rule as well as a Newton-Cotes quadrature of order 4 need an increment of the number of intervals in order to get reasonable results. For instance, composite Simpson's rule with 10 intervals gives an error of $E = 0.0127$ and with a discretization of $m = 50$ gives $E = 1.4525e-05$. This is not surprising due to the kind of sinusoidal behavior that this function shows. On the other hand, as we can see in the convergence plot, Gauss quadratures give a better result with less number of points. In the Lab class, composite trapezoidal rule was used to approximate the value of this integral but as it is shown in Figure 6(b), a large number of points need to be considered. As a result, the computational cost is by far the highest.

Integral 3 (Integral_3.m)

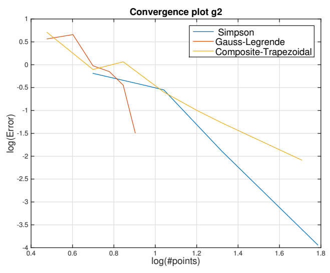
For this integral, we used composite Simpson's rule as a first approach. Again, we need to increase the number of intervals for the integration to obtain a good result. A Gauss-Legendre quadrature was also tested, giving a better result with less points, as expected. Looking at the plot of function $g3(x)$ we notice a change in the derivative of the function. Thus, same as for function $g2(x)$ a better approach should be to divide the function into intervals where the slope of the derivative is similar what allows to get a better approximation (Figure 7). In the script this method is implemented where we used different Gauss quadratures for the intervals.

[1] A Matlab function named *NewtonCotes* is provided. Depending on the chosen n, the function computes the integral using the trapezoidal rule, Simpson's rule, etc.

[2] A function named *GaussLegendre* is provided. It gives the points where the functions need to be evaluated as well as the weights for any interval.

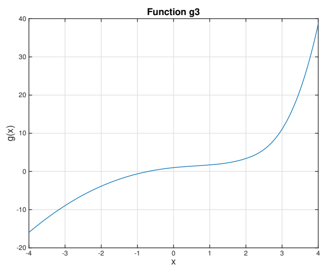


(a) Function $g2(x)$ is symmetric

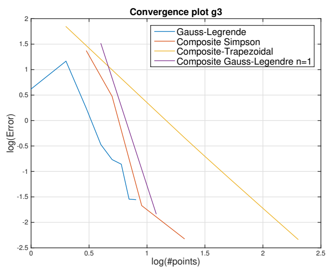


(b) Convergence

Figure 6: Plots for function $g2(x)$



(a) Function $g3(x)$



(b) Convergence

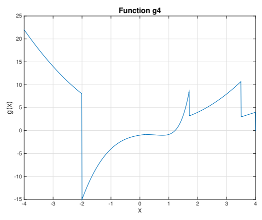
Figure 7: Plots for function $g3(x)$

Integral 4 (Integral_4.m)

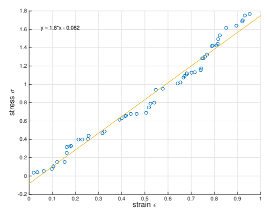
This function has discontinuities at abscissa $x = -2$, $x = 1.7$ and $x = 3.5$ which should be treated carefully in the integration (see Figure 8(a)). Therefore, 4 different subdomains are considered. In fact, between $x = -2$ and $x = 1.7$ we used the same approach as for previous integral, thus we actually had 5 subdomains. For the first interval, Simpson's rule was used (it looks like a 2nd order polynomial). For the last one, trapezoidal rule was used due to the shape of the function. In the middle intervals, Gass quadratures are considered as they use less points (see script).

Integral 5 (Integral_5.m)

The last function is defined as a set of points for stress and strain. After plotting (Figure 8 (b)), a linear trend in the data is observed. Thus, we obtained a linear fit using least squares and integrate the function. A Gauss quadrature with one point ($n = 0$) can be used to compute the value of the integral of the linear fit giving in this case $I = 0.741293$. Moreover, the trapezoidal rule can be used after joining the data points with lines and taking the distance between the points as the interval. This is the approach implemented in the Lab class but results in higher computational cost.



(a) Function $g4(x)$ plot



(b) Linear fit

Figure 8: Functions $g4(x)$ (a) and stress-strain plot with the linear fit