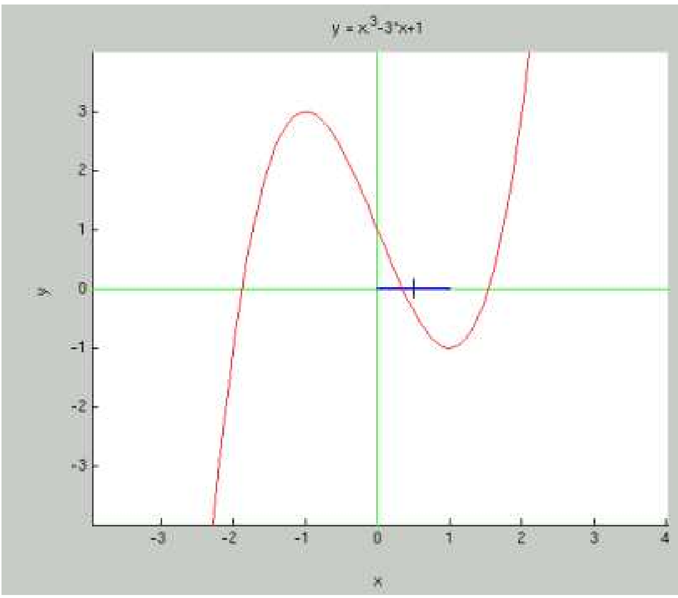
THE BISECTION ALGORITHM

COMPUTER SESSION D1



BACKGROUND

Natural science is about formulating equations (modeling) and then solving them (computation). This session will cover the Bisection algorithm which in a simple way solves (finds one or more roots) of the equation:

$$f(x) = 0 \tag{1}$$

where $f : \mathbb{R} \to \mathbb{R}$ is a Lipschitz continuous function.

For example, if you have an equation $3x = 2$, then you would move everything on the right hand side to the left hand side and get $3x - 2 = 0$. You would then define $f(x) = 3x - 2$ and could solve it using the Bisection algorithm.

The Bisection algorithm starts with an interval $I_0 = [a_0, b_0]$ so that $f(a_0)f(b_0) < 0$ (the signs of $f(a_0)$ and $f(b_0)$ must be opposite). It then splits the interval around the midpoint $\hat{x}_0$ into two subintervals and chooses the subinterval which still has opposite signs at the endpoints. This new interval is called $I_i = [a_i, b_i]$. The algorithm is then repeated from the beginning, but with I_i instead of I_0. It stops when $|a_i - b_i|$ is below a certain tolerance TOL. The result is the midpoint of the final interval $\hat{x}_i$ (alternatively it can return the final interval itself).

Before today's computer session, make sure that you understand and can answer the following questions.

Question 1

What does $f : \mathbb{R} \to \mathbb{R}$ mean? Does the bisection algorithm work for functions of several variables?

Question 2

What is a `while` loop?

PREPARATIONS

The session is divided into two parts. The first part involves experimenting in the Mathematics Laboratory and the second part involves writing your own implementation of the Bisection algorithm.

Bisection in the Mathematics Laboratory. Start Matlab.

If you are working on the computers of the School of Chemical Engineering at Chalmers, then download the file `startmath.m` to your Matlab work directory (if you have not done this already). This file is available on the web page of this session under Programs and templates. Then type `startmath` at the Matlab prompt. This command sets the search path to the directories where the Mathematics Laboratory is kept.

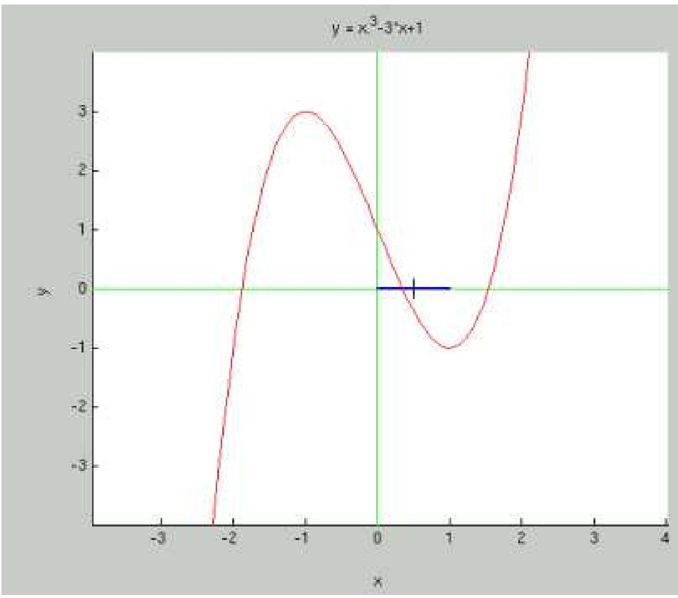
If you are working on another computer, then download the file `MathematicsLaboratory.zip` to your Matlab work directory. This file is available on the web page of this session under Programs and templates. Unzip the file, it should create a directory guis in your Matlab work directory. At the Matlab prompt, type: `addpath guis`. You are now ready to use the Mathematics Laboratory.

Keep your AMBS book with you and open at the relevant chapters.

Implementing the Bisection algorithm. Download the file: `bisect.m` to your Matlab work directory. This file is available on the web page of this session under *Programs and templates* http://www.phi.chalmers.se/bodysoul/sessions/d1/programs/.

Problems

Problem 1 - Bisection in the Mathematics Laboratory.



Note: don't spend too much time in the Mathematics Laboratory, only use it to understand how the algorithm works. It's more important to try to implement the algorithm yourself.

Give the command `open('RM+.fig')` (or possibly `open('RMplus')`) if this doen't work) to open the Road Map to the Mathematics Laboratory. Press the Bisection button to enter into the Bisection lab. Alternatively you may enter this lab directly from the matlab prompt by the command open('BISECT.fig').

Use the Bisection lab as follows:

Give $f(x)$, and a "return" to plot its graph (or choose one from the menu).

Look for a root (solution) of the equation $f(x) = 0$ to be solved, and give an x-value (somewhat) to the left of the root in the left $x =$ box below the plot window, followed by a "return" to have the corresponding $f(x)$ value computed and displayed.

Then give a x-value to the right of the root in the rightmost $x =$ box, again followed by a "return" to get the corresponding $f(x)$ value.

Check that the two $f(x)$ values just computed have opposite signs, indicating an $f(x) = 0$ value in between.

Now start the iteration process by computing and give the x-value of the midpoint of the given interval in the middle $x =$ box, and a "return" to have the corresponding $f(x)$ value computed and displayed. Then choose the left or right subinterval by pressing the left or right button, depending on where you think the root is. You determine this by checking the sign of $f(x)$ in the three present x points (or simply by looking at the graph, possible for you but not the computer). Watch how the current left or right x point is replaced by the previous midpoint, and how the previous midpoint is replaced by the midpoint of the choosen subinterval, and how all the corresponding $f(x)$ values are updated.

Next just repeat this procedure by choosing the left or right subinterval of the current interval, etc.

After a few iterations you may want to zoom in to see what is going on.

Also, note the list of left and right interval endpoints presented in the Matlab command window.

You may also automate the iteration process by giving a certain number of iterations, or a certain tolerance.

(1) As a first example, solve the equation $3x = 2$.
(2) Then solve the equation $x(1 + x)^2 = 1$.
(3) Find all roots of the equation $x^3 - 3x + 1 = 0$.
(4) How many iterations is required, starting with an interval of length 1, to have a root caught in an interval of length at most 0.01?
(5) What happens if the midpoint happens to be a root? Test!

(6) If there are roots in both subintervals, which one is chosen?
(7) Observing that the graph appears more and more linear as we zoom in on a root, it should be possible to compute a good expected value of the root from the last subinterval based on the size of $f(x)$ in the two endpoints. Derive a formula for this.
(8) Is the midpoint of the current interval the best point to make the cut into two subintervals?
(9) Pose further questions of your own problems and experiment!

Problem 2 - Implementing the Bisection algorithm.

(1) Now make your own $f(x) = 0$ solver by implementating the Bisection algorithm according to the specification in Bisection code shell (the file `bisect.m`). First replace the ?'s to get the code working. Then perform adequate tests to make sure the solver works as desired.

The Bisection algorithm in the code shell is implemented as a function `bisect(int, tol)` which takes a 2-element vector `int` specifying an interval $[a, b]$ and a tolerance `tol` as arguments. You need to define the function $f(x)$ representing the equation you want to solve in a separate file `f.m`. `bisect(int, tol)` then returns an approximate root (solution) of the equation within the given tolerance.

Example usage:

Define the function f in f.m as:

```
function y = f(x)
% f(x)
% Returns the sine of x.
    y = sin(x)
```

Then `x = bisect([1,4], 1e-7)` computes π to 7 decimals.

(2) You may want to add functionality to your bisection code as indicated (the add's), so it can handle also inconsistent input data. Consider also what happens if x (the computed midpoint) should be an exact root.
(3) If you have time, modify your implementation (or create a new function called `fbisect()` based on your existing `bisect()`) to take a function as an argument, and then use the function `feval()` to evaluate the function given as argument. The example code shell `fbisect.m` (also found under *Programs and templates*

http://www.phi.chalmers.se/bodysoul/sessions/d1/programs/) extends the original `bisect.m` with `feval()`.

If you don't do this, you need to redefine your function by modifying the file `f.m` every time you want to solve a different equation. If your implementation takes a function as an argument, you can define several functions `f.m`, `g.m` or `myfunction.m` and solve them by just giving a different argument.

Example usage of `fbisect(f, int, tol)`:

Define the function f in f.m as:

```
function y = f(x)
% f(x)
% Returns the sine of x.
    y = sin(x)
```

Then `x = fbisect('f', [1,4], 1e-7)` computes π to 7 decimals.

Define the function g in g.m as:

```
function y = g(x)
% g(x)
% Returns x^2 - 2, representing the equation x^2 = 2.
    y = x^2 - 2
```

Then `x = fbisect('g', [1,4], 1e-7)` computes $\sqrt(2)$ to 7 decimals.

SOLUTIONS

Make sure that you really try to solve each problem before looking at the solutions. Have you really tried to solve the problem or should you try again before looking at the solution?

The solutions are available on the web page of this session under *Solutions to problems*.

ABOUT

This Computer Session is part of the Body and Soul educational program. More information can be found at

`http://www.phi.chalmers.se/bodysoul/`

This Computer Session is maintained by Johan Jansson (johanjan@math.chalmers.se).