THE FIXED POINT ITERATION ALGORITHM

COMPUTER SESSION D2

BACKGROUND

We have previously looked at the Bisection algorithm for solving an equation $f(x) = 0$. We will now look at the Fixed Point Iteration algorithm for solving the same equation. This algorithm is commonly used in practice, for example, Newton's method is one of the most popular algorithms for solving equations, and it is a special case of the Fixed Point Iteration algorithm.

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

Question 1

How do you rewrite an arbitrary equation into $f(x) = 0$ form?

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 Fixed Point Iteration algorithm.

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.

PROBLEMS

Problem 1 - Fixed Point Iteration 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.

(1) Give the command `open('RM+.fig')` to open the Road Map to the Mathematics Laboratory, and press the Fixed P iter button to enter the Fixed Point iteration lab. Alternatively you may enter this lab directly from the matlab prompt by the command `open('FIXEDP.fig')`.

(2) Use the Fixed Point iteration lab as follows:

Give $g(x)$, and a "return" to plot its graph (or choose one from the menu). Note that also the "diagonal" $y = x$ is plotted.

Look for an x such that $g(x) = x$, that is an intersection of the graphs $y = g(x)$ and $y = x$, and give a nearby x-value in the x= box below the plot window, followed by a "return" to have the corresponding $g(x)$ value computed and displayed (on the y-axis but also on the x-axis, for conveniant comparison with the given x-value. Recall that we seek an x such that $g(x) = x$).

Next put the value of $g(x)$ into the x-box, that is redefine x as $g(x)$ by putting $x = g(x)$, manually (folled by a return or simpler by pressing the iterate button. Repeat!

You may also automate the iteration process by giving a certain number of iterations, or a certain tolerance (and a return in the corresponding edit box).

(3) As a first couple of examples, solve:

a) $0.3 + 0.3x = x$ b) $0.6 - 0.3x = x$ c) $1/(1 + x^2) = x$ (you have to write `./` and `.^` in the last example) Try starting from both sides of the solution, say from $x = 0$ and $x = 1$, respectively.

(4) Then try solving the equation $1 - x^3 = x$. If necessary, rewrite the equation, for example as $(1 - x^3 + kx)/(k + 1) = x$ (obtain by

adding kx to both sides and dividing by $k+1$), here with $k = 1$ or $k = 2$, say.
(5) Find both roots of the equation $x^2 + 2x - 1 = x$.
(6) Solve a) $2 - x^3 = 0$ b) $x^2 - 3 = 0$. Hint: Add kx to both sides and divide by k.
(7) Compare the number of iterations required to reach the root $x = 0$ for the equations a) $x/2 = x$ and b) $x^2 = x$, starting at $x = 0.4$, say.
(8) Pose further questions of your own and experiment!
(9) Now make your own $f(x) = 0$ solver by implementating the Fixed Point Iteration algorithm.

Problem 2 - Implementing the Fixed Point Iteration algorithm.

(1) Now make your own $f(x) = 0$ solver by implementating the Fixed Point Iteration algorithm according to the following specifications:

Write a function called `fixpoint(x0, tol)` which takes a starting guess `x0` and a tolerance `tol` as input arguments and gives an approximate solution as output. The function $f(x)$ representing the equation is assumed to be defined in `f.m`. The function header in Matlab could look something like this:

```
function x = fixpoint(x0, tol)
```

Example usage:

Define the function f in f.m as:

```
function y = f(x)
% f(x)
%
% Returns xˆ2 - 2, representing the equation xˆ2 = 2.
    y = xˆ2 - 2;
```

Then `x = fixpoint(1, 1e-7)` computes $\sqrt(2)$ with a tolerance of 1e-7.

Think about before you start implementing your function:

Look in the AMBS book in chapter 19.

Fixed Point Iteration is based on rewriting $f(x) = 0$ as $g(x) = x$, and then performing the iteration:

$$x_i = g(x_{i-1})$$

If we multiply by $-a$ and then add x to both sides of $f(x) = 0$, we get $x - a * f(x) = x$. This means we can define $g(x) = x - a * f(x)$. So we can write the iteration as:

$$x_i = x_{i-1} - a * f(x_{i-1})$$

a needs to be chosen so that the iteration converges. For the example above, $x^2 = 2$, $a = 0.1$ is sufficient.

To "iterate" is to do something several times. This is exactly what a "loop" does in programming. A `while` loop would be appropriate here, since we don't know in advance how many times we want to loop.

We need a *stopping condition* in the `while` loop so that we know when to stop the loop.

One way to measure how close we are to a solution of $f(x) = 0$ is to look at the *residual*. We simply put in our approximate solution x_i into $f(x)$, and if it is not a solution, $f(x_i)$ will not be 0. The closer we get to the solution, the smaller the residual will be.

We can define a variable `residual` in our Matlab function:

```
residual = f(x);
```

and check the absolute value of the residual (`abs(residual)`) in the stopping condition (and update the residual in the loop).

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).