Note on an efficient iterative method

by Ole Morten R.Isdahl

Solving equations or processes iteratively can often be a time consuming and power demanding processes. The following note describes the logic behind an algorithm originally developed to find the yield to maturity of bonds in the financial market. However, the method appeared to be very efficient, and to make an additional generalisation of the algorithm was of interest. This note also provide the source code for an example script.

The algortim is generalised to solve equations of the form

$$f(x, c_1, c_2, .., c_n) = 0 \tag{1}$$

where x is a variable and c is constants. The algorithm might be cascaded to include several constants as higher order variables and thus solving the equation in multiple dimensions simultaneously.

The algorithm

1. Generate a large coarse x-interval, certain to contain the true value, zero.
2. Calculate $f(x, c_1, c_2, .., c_n)$ for all x in the interval.
3. Find the x-value corresponding to the smallest solution larger than zero.
4. Assign that x-value as new upper limit for the interval
5. Assign the immidiate successor to the upper limit as the lower limit.
6. Repeat until wanted accuracy.
7. Solution = (upper limit + lower limit)/2

Example 1

Finding the square root of 2 numerically.

$$x = \sqrt{2} \tag{2}$$

Alternative form:

$$x^2 - 2 = 0 \tag{3}$$

1. Generate a coarse interval: 0, 1, 2, 3, 4
2. Calculate $f(x, c_1, c_2, .., c_n)$ for all x in the interval: -2,-1,2,7,14
3. Find the x-value corresponding to the smallest solution larger than zero.
4. 2
5. 1
6. Repeat until wanted accuracy. Iteration 1
7. Solution = (2 + 1)/2 = 1.5

Iteration 2: Solution = 1.375
Iteration 3: Solution = 1.40625
Iteration 4: Solution = 1.4140625
Iteration 5: Solution = 1.416015625
Iteration 6: Solution = 1.416015625
Iteration 7: Solution =1.4141845703125
Iteration 8: Solution = 1.414215087890625
Iteration 9: Solution = 1.414207458496094
Iteration 10: Solution = 1.414213180541992
Iteration 11 Solution = 1.414213657379150
Iteration 12 Solution = 1.414213538169861
Iteration 13 Solution = 1.414213567972183
Iteration 14 Solution = 1.414213560521603

Accuracy = 0.000000001
Course interval size: 5
Time to calcualte: < 1 ms using Intel(R) Core(TM) i7-3770 CPU @ 3.40 GHz
Note: Increasing the size of the course interval to 10 would reduce number of iteration to 8.

Computer-script (MATLAB)

```
% Iterative function solver
% Pre-filled out for calculating X^2-2 =0
clear
clc

int = 5;              % Size of interval
lower = 0;            % Intitial lower limit
upper = 4;            % Intitial lower limit
ac = 1*10^-9;         % Accuracy
noit = 0;             % Number of iterations


while true
    x = linspace(lower,upper,int);
    f = x.^2-2;
    upper = x(min(find(f>0)));
    lower = x(min(find(f>0))-1);

    if abs((f(min(find(f>0)))+f(min(find(f>0))-1)))/2 < ac
        break
    end
    noit = noit +1;
end

Solution = (upper+lower)/2
noit
```