

Finding roots of nonlinear equations

1 Introduction

This application note draws inspiration from the work of Huang et al. (see [YIPENG HUANG et al. 2017]).

The continuous Newton method is a powerful numerical algorithm for solving non-linear equations that has been gaining popularity due to its efficiency and robustness. However, its implementation on digital computers can be computationally expensive, especially for problems with a large number of variables.

This application note shows how to implement the continuous Newton method on an analogue computer using a few easy-to-reproduce examples to demonstrate its effectiveness. The examples are all one-dimensional problems but for more complex systems of equations with multiple variables in higher dimensions this approach can be used too. Detailed instructions are provided in the following, making the experiments easily reproducible. Overall, it is shown that analogue computing can be a valuable tool for solving non-linear equations. Hence they can also be used to solve many bigger problems such as solving non-linear partial differential equations via finite elements methods.

The classical Newton method starts with an initial guess and then iteratively refines the guess using

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \quad (1)$$

where $f(x)$ is the function whose root is being sought and $f'(x)$ is its derivative. However this method is not feasible for fast fluctuating functions and a wrong initial guess may lead to divergence.

In recent years, the continuous Newton method has emerged as an alternative to the classical Newton's method since it improves its sensitivity towards fluctuation. The method involves solving (a system of) ordinary differential equations. It is derived from the Newton's method by first introducing a damping factor to the equation. Therefore one starts by changing equation (1) to

$$x_{n+1} - x_n = -h \cdot \frac{f(x_n)}{f'(x_n)}. \quad (2)$$

Then, via $h \rightarrow 0$ one obtains the differential equation

$$\dot{x}(t) = -\frac{f(x(t))}{f'(x(t))}. \quad (3)$$

Analogously in the multi-dimensional case one can conclude

$$\dot{x}(t) = -[J(x(t))]^{-1}f(x(t)) \quad (4)$$

where $J(x)$ is the Jacobian matrix of $f(x)$. This is referred to as the continuous Newton method. It offers an efficient and robust alternative to the classical Newton's Method for solving non-linear equations, especially for problems with a large number of variables.

2 Analog Implementation for linear and quadratic function

Two easily implemented toy problems are linear and quadratic functions. For linear equations of the form $f(x) = ax + b$, where $a, b \in \mathbb{R}^{>0}$,

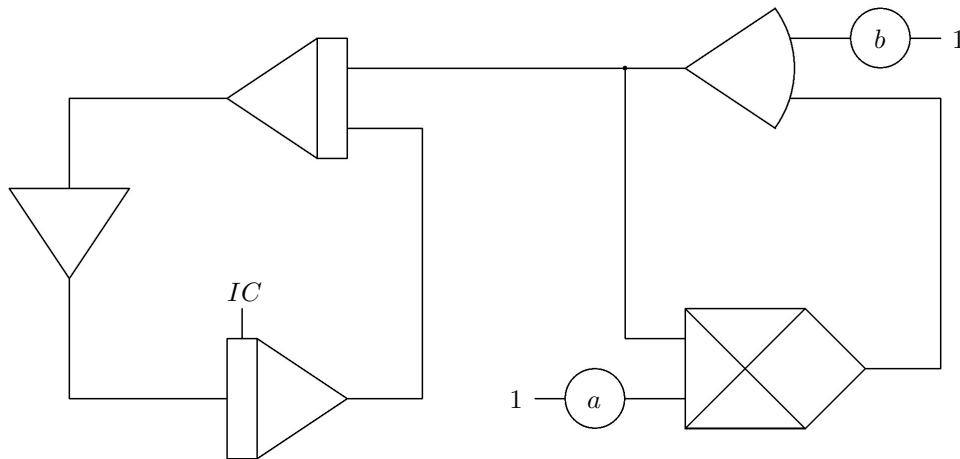


Figure 1: Analog root finder for $f(x) = ax + b$

equation (3) leads to

$$\dot{x}(t) = x + \frac{b}{a} \quad (5)$$

and can be implemented on an analog computer just as shown in figure 1.

The divider used in figure 1 comes with a restriction: The denominator cannot be negative. To fix this problem for the next example this setup has been improved with the help of two additional comparators and summers. The improved divider can be seen in figure 2.

With this upgrade a more interesting example can be solved on analog computers such as a THAT. Here, roots for the set of quadratic functions of the form $f(x) = (x + a)^2 - b$ have been determined by implementing

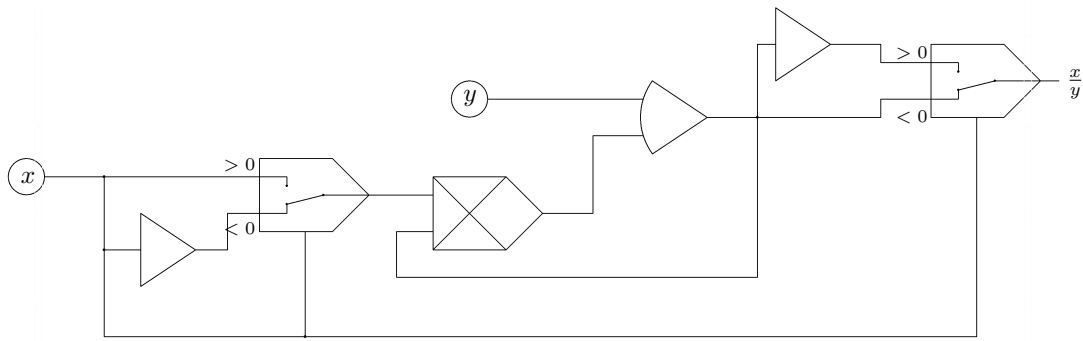


Figure 2: Improved Divider. Here the denominator y can be negative too.

the following ODE

$$\dot{x}(t) = \frac{(x + a)^2 - b}{2(x + a)}.$$

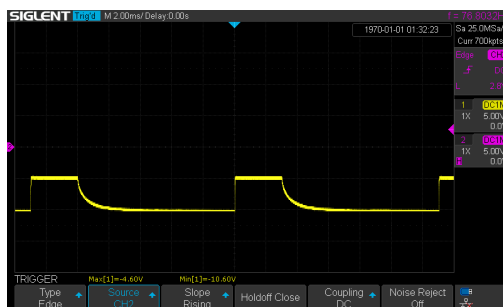
The circuit for implementing this setup on an analog computer is shown in figure 4. One can see the large impact of the initial value x_0 .

Figure 3 emphasizes the importance of properly chosen initial conditions. In the case of $f(x) = (x + 0.5)^2 - 0.25$ one can see that for $IC > -0.5$ equation (3) will converge against the root 0. If, however, $IC < -0.5$ the root $x = -1$ is found. One should be aware of the fact that a “hard boundary” for the initial condition is not always the mid-point between two neighbored roots. Instead, most of the time the next extremum has a great impact on the convergence of the method.

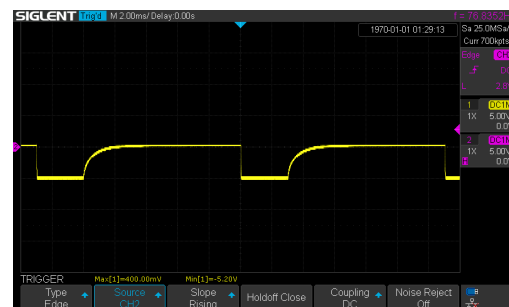
3 An example of a cubic function

The following example has been implemented on a Model-1 which features dedicated dividers. The corresponding analog circuit diagram is shown in figure 6. The results can be seen in figure 5. Both implemented scenarios demonstrate a function with a local maximum below the x-axis but with different initial conditions.

Again, a bad initial guess will lead the differential equation to approach an extremum, resulting in a division by zero and thus an overload. (a) illustrates that there are no issues when starting to the right of the local minimum (below the x-axis), as the differential equation's solution moves towards and ultimately converges to the root. On the other hand, if one starts with a poor initial guess, the system behaves as depicted in (c), adhering to the expected overload limits at ± 1 . This leads to oscillations around the extremum point, as depicted in (b).



(a) IC= 0.46



(b) IC= 0.52

Figure 3: Impact of the initial condition on the continuous newton method

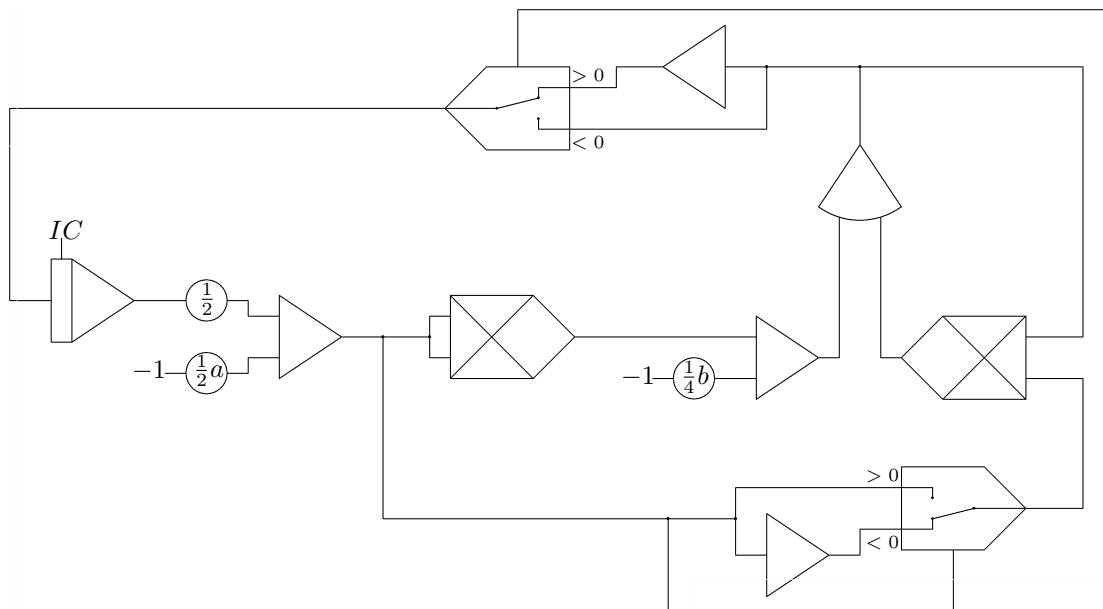
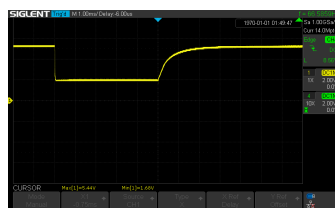


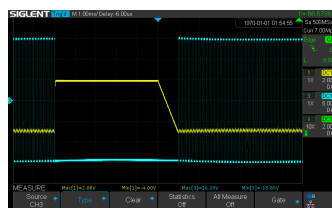
Figure 4: Analog root finder for $f(x) = (x - a)^2 + b$.

References

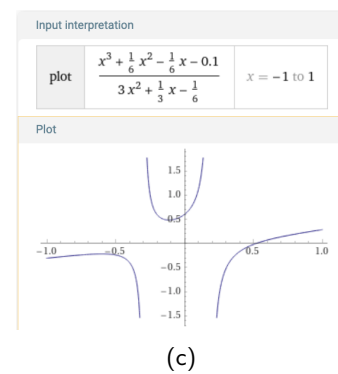
[YIPENG HUANG et al. 2017] YIPENG HUANG, NING GUO, MIN-GOO SEOK, YANNIS TSIVIDIS, KYLE MANDLI, SIMHA SETHUMADHAVAN, "Hybrid Analog-Digital Solution of Nonlinear Partial Differential Equations", in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, ACM, 2017



(a)



(b)



(c)

Figure 5: The first two images in the series display the analog results for a good and a poor initial guess, respectively. The third image illustrates the theoretical quotient of f and f' .

