Finding roots of nonlinear equations

This application note draws inspiration from the work of Huang et al. [1].
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 by the example of a few easy-to-reproduce examples to demonstrate its effectiveness. The examples of this paper show only one-dimensional problems but for more complex systems of equations with multiple variables in higher dimensions this approach can be used too. This Note provides detailed instructions, 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 e.g. solving non-linear partial differential equations via finite elements method.
The classical Newton's method starts with an initial guess and then iteratively refines the guess by using the following equation

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

where $f(x)$ is the function whose root is being sought and $f'(x)$ its derivative. However this method isn't feaseble 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. and derived from the Newton's method by first introducing a damping factor to the equation. Therefore one starty by turning equation (1) to

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

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

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

Analoguously in the multi-dimensional case one can conclude

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

where $J(x)$ is the Jacobian matrix of $f(x)$ and which is referred to as the continuous newton method. The continuous Newton method 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.

Analoge Implementation for linear and quadratic function

Two easy to implement toy problems are linear and quadratic functions. For linear equations of the form $f(x) = ax + b$, where $a, b \in \mathbb{R}^{>0}$, equation (3) leads to:

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

and can be implemented on an analog computer just as shown in figure 1. However the in figure 1 implemented divider comes by construction with a restriction. The denominator can't be negative. To fix this problem for the next example this setup has been improved with the help of two additional comperators and summers. The improved divider can be seen in figure 2.

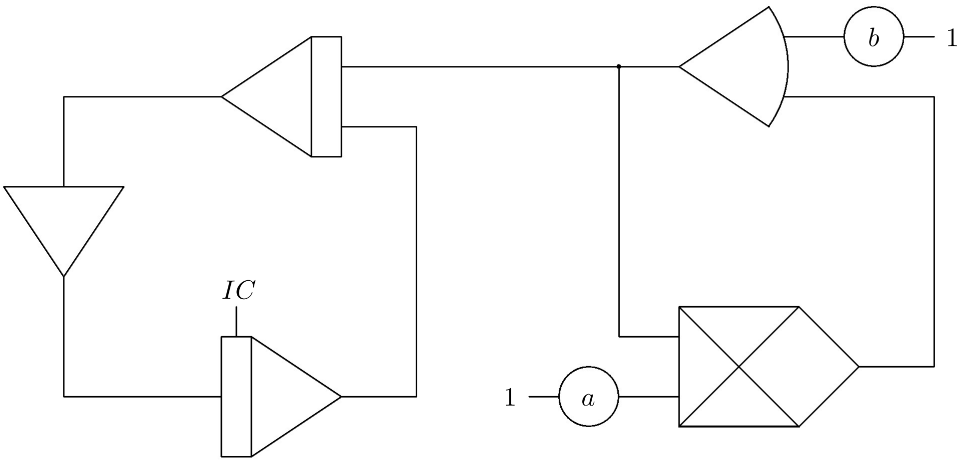


Figure 1: analog root finder for $f(x) = ax + b$

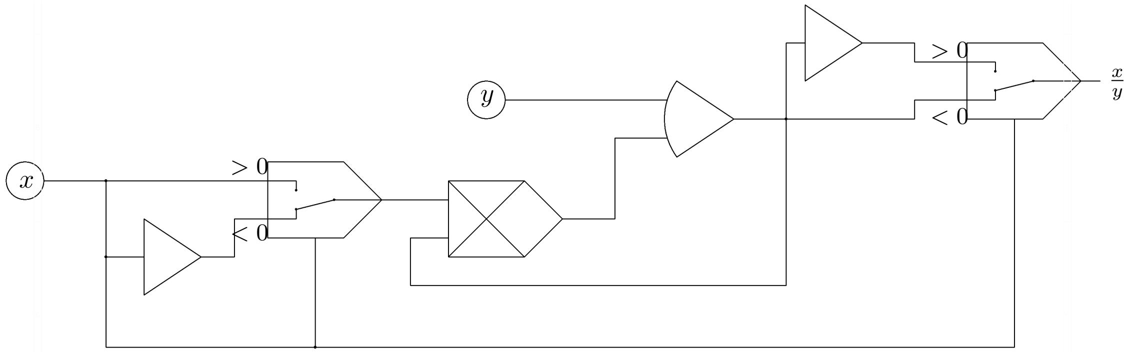
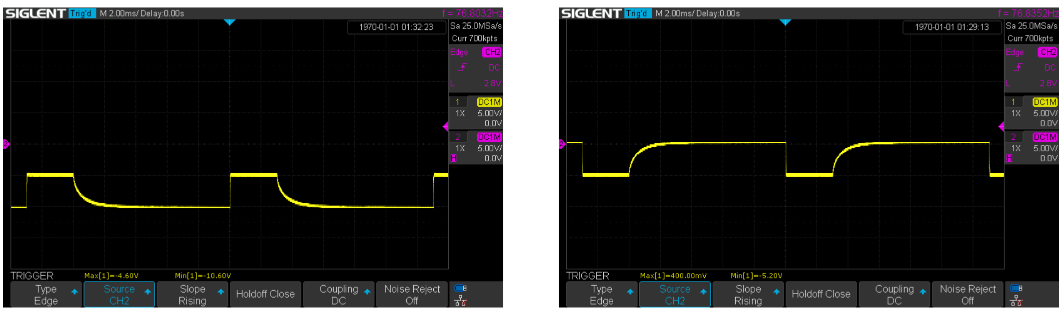


Figure 2: Improved Divider. Here the denominator y can be negative too.

With this upgrade an a bit more interesting example can be created on analog computers without a dedicated divider such as a THAT. This time for the set of quadratic functions of the form $f(x) = (x+a)^2 - b$ roots have been determined by implementing the following ODE

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

The circut for implementing this setup on an analog computer is shown in figure 4. At this example one can see how much of an impact the initial value x_0 has got. Figure 3 emphasizes the importance of a proper chosen initial condition. For the example $f(x) = (x+0.5)^2 - 0.25$ one can see, that for IC> -0.5 equation (3) will converge against the root 0 and if IC< -0.5 the root $x = -1$ is found. One may be aware of the fact that the "hard border" for the initial condition is not always the middle between the two neighboored roots, instead most of the time the next extrema has a great impact on the convergence of the algorithm.



(a) IC$= 0.46$ (b) IC$= 0.52$

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

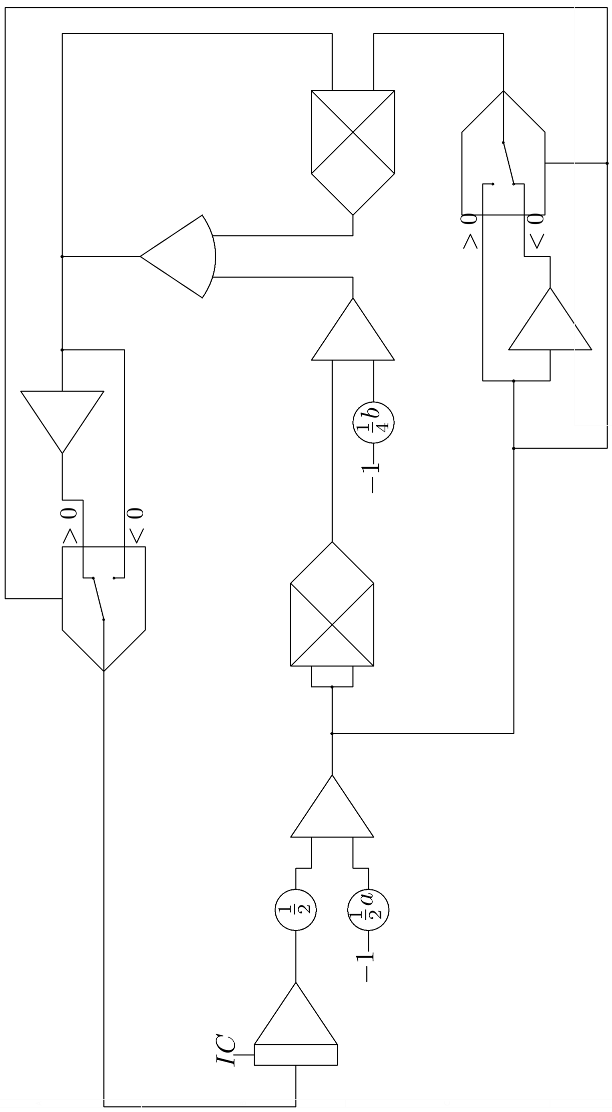
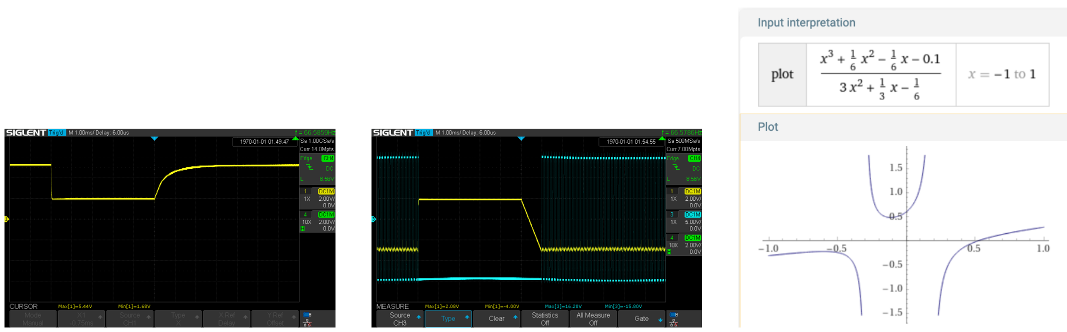


Figure 4: analog root finder for $f(x) = (x - a)^2 + b$.

An example for a qubic function

This example has been implemented on a model 1 with dedicated divider. The analog circuit diagram corresponding to this example is shown in Figure 6. The results of the analog implementation 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 the extremum, resulting in the observed behavior arising from division by zero. (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) (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'.
.

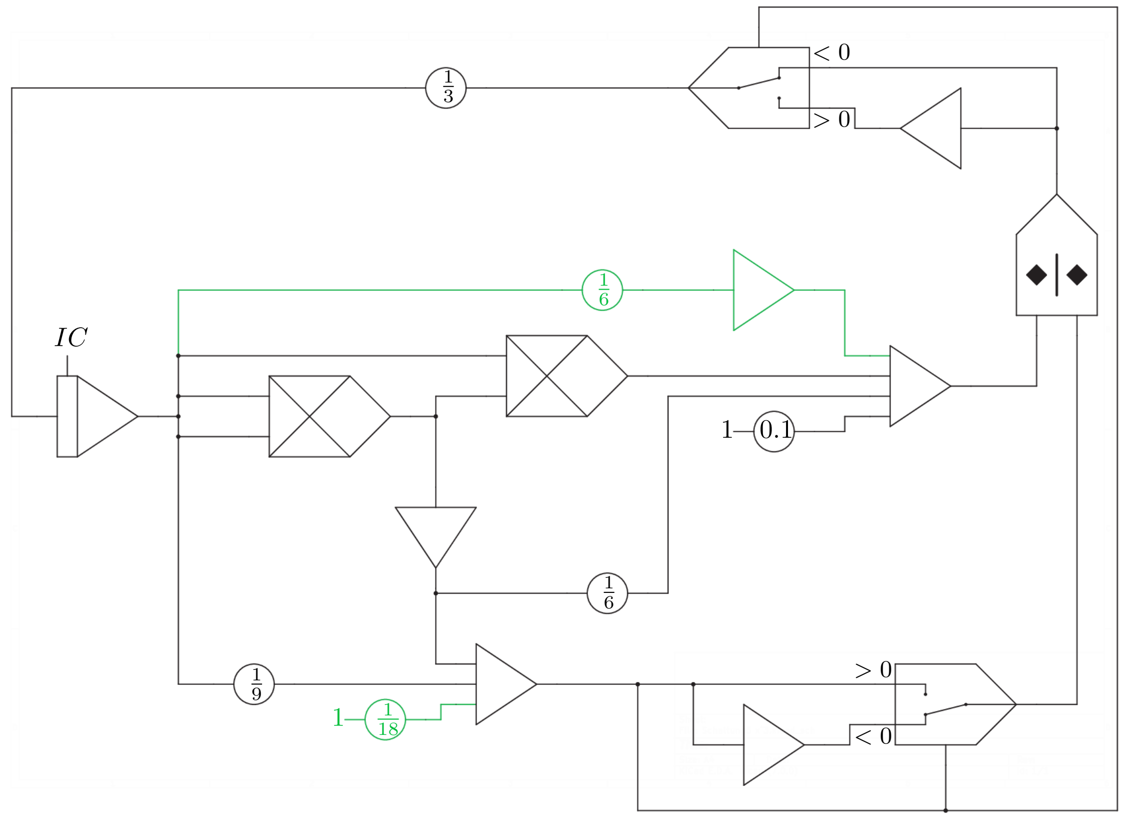


Figure 6: analog root finder for $f(x) = x^3 + \frac{1}{6}(x^2 - x) - 0.1$. By ommiting the green wires, one can ommit the linear term.

References

[1] Yipeng Huang et al. "Hybrid Analog-digital Solution of Nonlinear Partial Differential Equations". In: *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*. MICRO-50 '17. New York, NY, USA: ACM, p. 665678. ISBN: 978-1-4503-4952-9. DOI: 10.1145/3123939.3124550. URL: http://doi.acm.org/10.1145/3123939.3124550.