A REVIEW OF METHODS FOR RESOLVING DELAY-FREE LOOPS

Jatin Chowdhury

Center for Computer Research in Music and Acoustics
Stanford University
Stanford, CA
jatin@ccrma.stanford.edu

ABSTRACT

Systems containing delay-free loops are common in audio signal processing. We present a simple system containing a delay-free loop, and show that the system is non-computable by traditional means. We then outline several methods for resolving the delay-free loop, including iterative methods, explicit methods, and purely discrete time methods. Finally, we compare the proposed methods, and provide recommendations for signal processing engineers.

1. BACKGROUND

Start by considering the following continuous-time system shown in fig. 1, where $H(s)$ is some linear system, G is a purely linear gain, and f_{NL} is some one-to-one nonlinear function. For simplicity, in the examples below, we choose $f_{NL}(x) = \tanh(x)$, and $H(s)$ to be a first order lowpass filter, of the form $H(s) = \frac{w_c}{s+w_c}$, where w_c corresponds to the filter cutoff frequency. These types of systems occur rather frequently in physical modelling (see [1, 2, 3]); this example specifically is inspired by a guitar amplifier feedback model proposed by Sullivan [4].

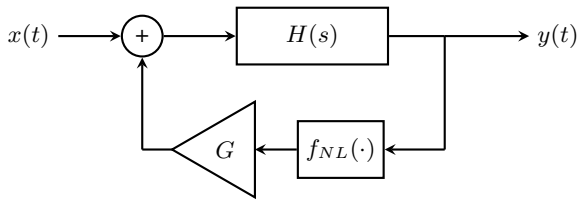

Figure 1: *Continuous time system with a delay free loop.*

1.1. Non-Computability

The general issue with delay-free loops in digital signal processing is that they lead to non-computable systems. As an example, let us attempt to naively construct a digital model of the system shown in fig. 1. First, we can digitize the linear system $H(s)$ as a digital filter $H(z)$ using some discretization scheme (often Backwards Euler or trapezoidal rule). Let us define $c[n]$ as the input to $H(z)$, and assume that $H(z)$ is a generic, first-order, IIR filter, with filter coefficients b_n and a_n; we can now compute the output $y[n]$ as follows.

$$y[n] = b_0 c[n] + b_1 c[n-1] - a_1 y[n-1] \tag{1}$$

However, we know that ,

$$c[n] = x[n] + G f_{NL}(y[n]) \tag{2}$$

so we can re-write eq. (1) as follows.

$$\begin{aligned} y[n] = b_0(x[n] + G f_{NL}(y[n])) \\ + b_1(x[n-1] + G f_{NL}(y[n-1])) - a_1 y[n-1] \end{aligned} \tag{3}$$

Note that both $y[n]$ and $f_{NL}(y[n])$ appear in eq. (3), meaning that an explicit solution for $y[n]$ cannot be computed directly. The simplest solution is to simply add a delay unit to the feedback path, however this changes the behavior of the system, specifically the frequency response. The following sections will outline methods for resolving the delay-free loop seen in this example in a mathematically correct fashion, providing results and discussion for each method.

2. ITERATIVE METHODS: NEWTON-RAPHSON

Iterative methods are perhaps the most commonly used method for resolving delay-free loops, and the Newton-Raphson method is perhaps the most commonly used iterative method. The Newton-Raphson method can be used for solving ordinary differential equations (ODEs) of the form.

$$\mathbf{F}(\mathbf{x}) = 0 \tag{4}$$

Below we will derive an ODE of this form for our example system, and show the resulting simulation using a Newton-Raphson solver.

2.1. Setting up the ODE

Recall from our definition of $H(s)$.

$$Y(s) = C(s)\frac{w_c}{s + w_c} \tag{5}$$

Cross-multiplying we see that,

$$sY(s) + w_c Y(s) = w_c C(s) \tag{6}$$

then using properties of the Laplace transform, we see that.

$$\dot{y}(t) + w_c y(t) = w_c c(t) \tag{7}$$

Finally, using the continuous-time version of eq. (2), we can solve for the time derivative of the output signal.

$$\dot{y}(t) = g(x(t), y(t)) = w_c(x(t) + G f_{NL}(y(t)) - y(t)) \tag{8}$$

or in the digital domain.

$$\dot{y}[n] = g(x[n], y[n]) = w_c(x[n] + G f_{NL}(y[n]) - y[n]) \tag{9}$$

Next we recall the trapezoidal rule for integration,

$$y[n] = y[n-1] + \frac{T}{2}(\dot{y}[n] + \dot{y}[n-1]) \tag{10}$$

where T is the sample period. Finally, we set up our equation in the form of eq. (4).

$$f(y) = y - y[n-1] - \frac{T}{2}(g(x[n], y) + g(x[n-1], y[n-1])) = 0 \tag{11}$$

2.2. Newton-Raphson Iteration

We can now attempt to solve this ODE iteratively, using Newton-Raphson iteration. As defined in [5], once we have an ODE in the form of eq. (4), we can find a solution iteratively using the following update equation,

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{J}_{\mathbf{F}}^{-1}(\mathbf{x})\mathbf{F}(\mathbf{x}) \tag{12}$$

where $\mathbf{J}_{\mathbf{F}}$ denotes the Jacobian of the function $\mathbf{F}$. Once the difference between consecutive iterations $\Delta = \mathbf{x}_{k+1} - \mathbf{x}_k$ has reached a sufficiently small value, the solution can be said to have converged. For our example system, we have.

$$f'(y) = 1 - \frac{T}{2}g'(x[n], y) = 1 - \frac{w_c T}{2}(G f'_{NL}(y) - 1) \tag{13}$$

We can then derive the Newton-Raphson iteration as follows.

$$y[n]_{k+1} = y[n]_k - \frac{y - y[n-1] - \frac{T}{2}(g(x[n], y) + g(x[n-1], y[n-1]))}{1 - \frac{w_c T}{2}(G f'_{NL}(y) - 1)} \tag{14}$$

2.3. Discussion

The Newton-Raphson method is widely used for computing nonlinear delay-free loops, largely because of its accuracy. In particular, as the sample rate grows to be sufficiently large, the Newton-Raphson method can achieve arbitrarily high accuracy (up to the numerical precision of the system), although high accuracy simulation may require a large number of iterations for complex systems. Drawbacks include the high computational cost of running many iterations, which is made worse for high dimensional ODEs, since computing $\mathbf{J}_{\mathbf{F}}^{-1}$ requires a matrix inversion at every iteration. That said, the performance of the Newton-Raphson method (and nearly all iterative methods) can be greatly improved through the use of a intelligently chosen initial estimate for the solution.

3. EXPLICIT SOLVERS: RUNGE-KUTTA METHOD

The Runge-Kutta method is a standard numerical technique for solving ODEs, and the 4th-order Runge-Kutta method (RK4), is probably the most commonly used single-step technique for solving ODEs in real-time audio signal processing. The use of RK4 specifically for resolving systems with delay-free loops is outlined in detail in [3]. For our example system, we can resolve the delay-free loop using RK4 as follows: First, we form the same differential equation expressed by eq. (9). Then, we compute future output samples as follows.

$$\begin{aligned}
k_1 &= T g\left(x[n-1], y[n-1]\right) \\
k_2 &= T g\left(x\left[n - \frac{1}{2}\right], y[n-1] + \frac{k_1}{2}\right) \\
k_3 &= T g\left(x\left[n - \frac{1}{2}\right], y[n-1] + \frac{k_2}{2}\right) \\
k_4 &= T g\left(x[n], y[n-1] + k_3\right) \\
y[n] &= y[n-1] + \frac{k1}{6} + \frac{k2}{3} + \frac{k3}{3} + \frac{k4}{6}
\end{aligned} \tag{15}$$

Figure 2 shows a comparison of two simulations of the system shown in fig. 1, using the Newton-Raphson and Runge-Kutta methods.

3.1. Discussion

Comparing the relative efficiencies of the Runge-Kutta method and the Newton-Raphson method is difficult, since the Newton-Raphson method can be faster when it converges quickly, but it could also be much slower for instances when it takes many iterations to converge. That said, the Runge-Kutta method typically has a much faster maximum computation time. It is also possible to compute the maximum possible error for an ODE solution found using the Runge-Kutta method, for more details see [5].

4. DISCRETE-TIME METHOD

While the Newton-Raphson and Runge-Kutta methods rely on the the continuous-time formulation of an ODE describing the system, [6] and [7] describe a method for resolving delay-free loops of systems analyzed solely in discrete-time. First, we note that the output of any digital filter $H(z)$, can be written as,

$$y[n] = h_0 x[n] + \mathcal{H}_n \tag{16}$$

where h_0 is the first sample of the filter's impulse response, and $\mathcal{H}_n$ is the output of the filter for zero input with it's given state at sample time n. With this in mind, we can re-write the filter equations for our example system as follows.

$$y[n] = h_0(x[n] + G f_{NL}(y[n])) + \mathcal{H}_n \tag{17}$$

For certain nonlinear functions f_{NL}, this output can be computed directly, however, in the general case, an interative solution may be needed. For example, we could use the Newton-Raphson iteration by defining a function $f(y)$.

$$f(y) = y[n] - h_0(x[n] + G f_{NL}(y[n])) - \mathcal{H}_n = 0 \tag{18}$$

Then the Newton-Raphson iteration could be calculated as.

$$y_{k+1} = y_k - \frac{f(y)}{f'(y)} = y_k - \frac{y_k - h_0(x[n] + G f_{NL}(y_k)) - \mathcal{H}_n}{1 - h_0 G f'_{NL}(y_k)} \tag{19}$$

A comparison of simulations of the example system using the pure Newton-Raphson method and the discrete-time method described here is shown in fig. 3.

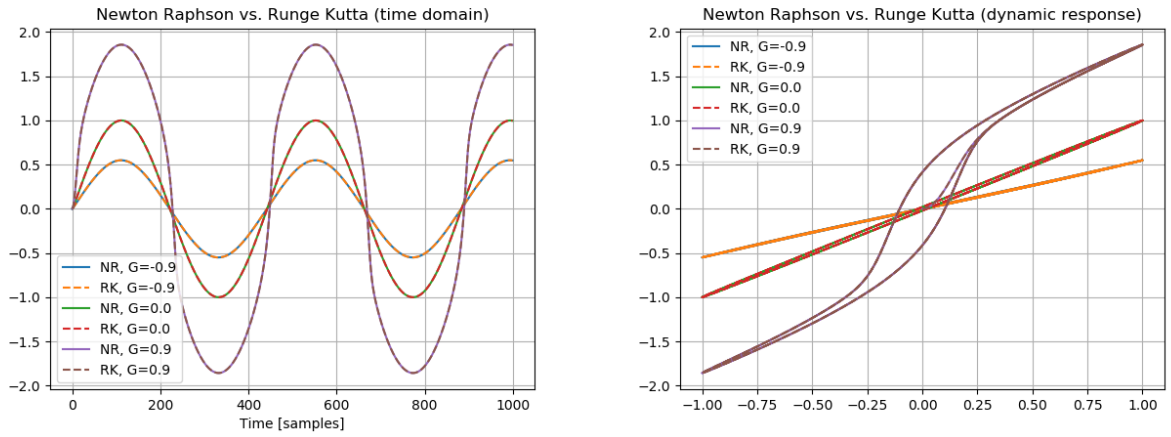

Figure 2: *Comparison of simulation results between Newton-Raphson (NR) and Runge-Kutta (RK) methods.* $f_{NL}(x) = \tanh(x)$, $w_c = 2\pi(5000\ Hz)$. *Input is a 100 Hz, unit amplitude sine wave.*

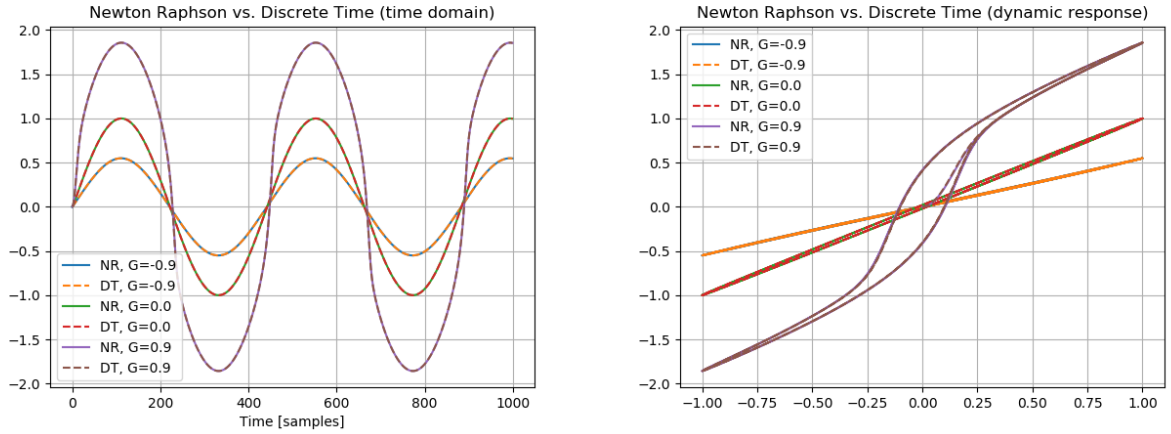

Figure 3: *Comparison of simulation results between Newton-Raphson (NR) and Discrete Time (DT) methods. As in fig. 2,* $f_{NL}(x) = \tanh(x)$, $w_c = 2\pi(5000\ Hz)$. *Input is a 100 Hz, unit amplitude sine wave.*

4.1. Discussion

The primary advantage of the discrete time method is the fact that it does not rely on the continuous time ODE, meaning that once a solution is found for a certain DSP architecture, the same solution can be applied to all other systems sharing the same architecture. In terms of performance, this method performs comparably to whatever iterative method is chosen for resolving the nonlinearity.

5. DISCUSSION

Each of the above methods can be used to correctly resolve delay-free loops in signal processing systems. For virtual analog systems, it is often necessary to derive a continuous-time ordinary differential equation defining the system, meaning that these types of systems naturally lend themselves to the use of the Newton-Raphson and Runge-Kutta methods. In situations where iterative methods converge quickly, they are often more efficient and more accurate than the Runge-Kutta method, and should be preferred. For purely digital systems, forming continuous-time equations often adds unnecessary complexity, so the discrete-time method is often simpler, more flexible, and reasonably efficient, compared to the other methods.

6. CONCLUSION

We have provided an example signal processing system containing a delay-free loop, and outlined several known methods for correctly resolving the delay-free loop, including the Newton-Raphson, Runge-Kutta, and discrete-time methods. We have discussed some of the advantages and disadvantages of each method, and provided recommendations for signal processing engineers when choosing a method to use for a given system.

7. REFERENCES

[1] G. Borin, Giovanni De Poli, and Davide Rocchesso, "Elimination of delay-free loops in discrete-time models of nonlinear acoustic systems," *Speech and Audio Processing, IEEE Transactions on*, vol. 8, pp. 597 – 605, Oct. 2000.

[2] J. Szczupak and S. Mitra, "Detection, location, and removal of delay-free loops in digital filter configurations," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 23, no. 6, pp. 558–562, 1975.

[3] David Medine, "Dynamical systems for audio synthesis: Embracing nonlinearities and delay-free loops," *Applied Sciences*, vol. 6, pp. 134, May 2016.

[4] Charles R. Sullivan, "Extending the Karplus-Strong algorithm to synthesize electric guitar timbres with distortion and feedback," *Computer Music Journal*, vol. 14, pp. 26, 1990.

[5] D.T. Yeh, *Digital Implementation of Musical Distortion Circuits by Analysis and Simulation*, Ph.D. thesis, Stanford Univeristy, June 2009.

[6] Dave Berners and Jonathan S. Abel, "Discrete-time implementation of arbitrary delay-free feedback networks," in *Audio Engineering Society Convention 141*, Sept. 2016.

[7] Vadim Zavalishin, "Preserving the LTI system topology in s- to z-plane transforms," 2008.