The Reaktor Core tutorial to accompany "Computational optimization of nonlinear zero-delay feedback by second-order piecewise approximation"

Vadim Zavalishin*

November 7, 2008

WARNING. This tutorial is not a beginner one. The basic knowledge of DSP theory as well as a good command of Reaktor Core is assumed.

Also, this is a tutorial and not a walkthrough. Be prepared to that a good amount of the material will be available only in the form of hints, with corresponding exercises.

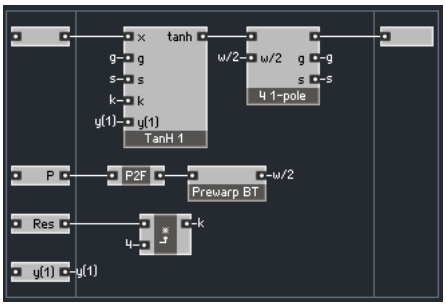

Figure 1: Ladder filter.

1 Introduction

The article "Computational optimization of nonlinear zero-delay feedback by second-order piecewise approximation", which this tutorial is supposed to accompany, describes a small set of DSP techniques for the design of discrete-time LTI systems. The purpose of this tutorial is to provide examples of how these techniques can be used within the Reaktor Core environment.

Using Reaktor to implement the algorithms that create the approximations can be a quite challenging task. While this could be a good excercise in Reaktor techniques, it probably doesn't make too much sense otherwise. Therefore in this tutorial we will restrict ourselves to the implementation of the filters described in the article. As the nonlinear function we will use the tanh approximation built in the article.

2 Ladder Filter

As the basis for the implementation we will use the implementation of the ladder filter built in *Reaktor Core tutorial to accompany "Preserving the LTI system topology in s- to z-plane transforms"*. The version which uses the Newton-Raphson iteration is probably the best starting point, as we essentially only need to replace the *NR* macro computing the Newton-Raphson iterations by another macro which explicitly implements the formulas Eq. 16 and Eq. 20b[1] (Fig. 1).

The *y(1)* input is providing the signal for $y_{0.5}$ (the notation is explained if we remember that $x_{0.5} = 1$).

*Native Instruments GmbH

[1]The Eq. prefix denotes the equations from the article.

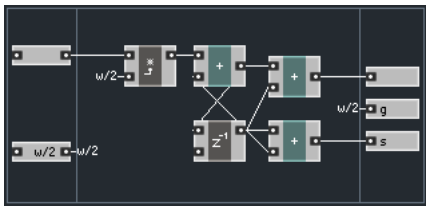

Figure 2: Bilinear integrator.

Questions and exercises

1. Implement the contents of the *TanH1* macro in Fig. 1.

2. Implement the complete structure in Fig. 1. Compare its sound at $y(1) = 0.78$ to the sound of the Newton-Raphson ladder filter implementation.

3 State-Variable Filter

The state-variable filter can also be based on the one from *Reaktor Core tutorial to accompany "Preserving the LTI system topology in s- to z-plane transforms"*. This time we need to make a little bit more modifications.

Since the integrators in the original filter output the 'accumulated' values of g and s, and the formulas Eq. 16 and Eq. 24 use the 'local' values, we need to modify the integrator structure accordingly (Fig. 2).

Now we can, similarly to the ladder filter, use a macro which implements the mentioned solution formulas (Fig. 3).

Questions and exercises

1. Explain the modifications to the bilinear integrator in Fig. 2.

2. Implement the contents of the *SolveNL* macro in Fig. 3.

3. Implement the complete structure in Fig. 1. Compare its sound at low signal levels, low resonance settings and $y(1) \in [0.5, 0.8]$ to the sound of the linear state-variable filter implementation.

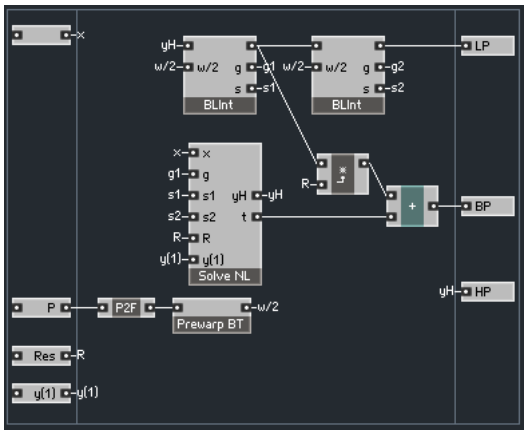

Figure 3: State-variable filter.

4 Homework

Questions and exercises

1. Design and implement your own filter or effect using the tanh nonlinearity in the feedback.

2. Implement a system with another nonlinearity (you'll probably need some other software besides Reaktor to build the approximation).

Acknowledgements

REAKTOR® and *REAKTOR Core Technology*® are registered trademarks of Native Instruments GmbH.