Experiments with the Nonlinear Stable Spline Kernel for System Identification

Gordon Davidson
student number: 86867892
Department of Computer Science
University of British Columbia
gordonwd@telus.net

Justin Reiher
student number: 37291151
Department of Computer Science
University of British Columbia
reiher@cs.ubc.ca

Jocelyn Minns
student number: 10658152
Department of Computer Science
University of British Columbia
jminns@cs.ubc.ca

Abstract

Recent work has introduced the use of kernels from the machine learning community to the field of system identification. In particular, the stable spline kernel and nonlinear stable spline kernels were shown to have the desired stablity properties for systems. The recent work introduced the kernels and their mathematical properties, but experimental work focussed on the comparison with the traditional parametric methods, and under a limited range of conditions. In this work we conduct a set of simulation experiments to characterize the accuracy of the stable spline and nonlinear stable spline kernels, under various conditions of input, noise, and nonlinearity. The results show that in the presence of high noise the SS is a better kernel to capture global response structure whereas the NLSS is well suited at capturing nonlinear behaviour in the absence of noise. The results also show that in the presence of moderate to high noise strength the NLSS starts to behave like the SS kernel in the estimate of the impulse response.

1 Introduction

A dynamic system takes an input (typically a function of time, either discrete samples or continuous), and produces a corresponding time-varying output according to some function or model. Many phenomenon in the sciences and engineering can be thought of as dynamic systems, and examples range from small mechanical devices to industrial plants to global weather systems. Very often in either the study of a natural system or in the analysis of an engineered system, it is desired to be able to model the behaviour of some given system from an input-output perspective. System identification refers to the inverse problem of constructing a model of a dynamic system given some known inputs and the observed, noisy output data that were produced. An important engineering application of system identification is in the control of an automated plant, where the model is used to be able predict outputs given certain inputs for the purposes of knowing how to acheive the desired behaviour. The field of system identification is a very established field with its own history, set of algorithms, software packages, etc. However there is definitely an overlap with the goals of machine learning - ie. learning a model from examples - and recently there has been interest in applying techniques from the machine learning community to the problem.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

System identification is traditionally done by estimating the parameters of a model with a fixed structure [3]. This can be posed as a classic regularized least squares problem, assuming the model order is known. However, assumptions about the model order have large implications on how the system performs, and can ultimately lead to very different control designs. Since the model order is often not known, there have been attempts to control the complexity of the model by incorporating penalty terms such as the Akaike information criterion (AIC). However these have been found to produce unsatisfactory results, particularly with noisy data [6].

In order to form an estimate the model without making assumptions about its structure, some non-parameteric, kernel-based approaches to system identification have been introduced from the field of machine learning [7, 1, 5]. The basic idea uses the representer theorem of kernels, so that the function modeling the system can be expressed as a weighted sum of kernel evaluations, at the training examples and at the desired evaluation points (test examples) of the function. The coefficients of the weighted sum are the solution to the kernelized, regularization problem. In addition, the model order is essentially determined by a decay rate that is given by a hyper-parameter of the kernel. This hyper-parameter is determined from the data using the empirical Bayes approach, in place of explicit assumptions about the model order.

Kernels for both linear and nonlinear system identification were introduced in [5]. The stable spline and nonlinear stable spline kernels were analyzed to show that they satisfy requirements on stability for system identification. However, the work presented limited experimental results on the performance. In this work we conduct a set of simulation experiments to characterize the accuracy of the stable spline and nonlinear stable spline kernels, under various conditions of input, noise, and nonlinearity. Results of this type of investigation are important for understanding the how the kernels may perform in practice.

The paper is organized as follows: Section 2 presents a brief background on system identification and describes the kernel method and the stable spline and nonlinear stable spline kernel. Section 3 describes the simulation experiment methodology. Section 4 shows results and comparison between the kernels. Section 5 is a discussion on the results, conclusions and future work.

2 Background

System identification is about finding a function which maps present and past inputs, $u(t), u(t-1), u(t-2), ...$, and past outputs $y(t-1), y(t-2), ..$, to a current output value $y(t)$. Here, discrete time is assumed, so the variable t is a time index. A property that the system producing y from u must satisfy is that if a bounded input signal u is applied, the output signal y is also bounded. Furthermore the system is assumed to be time-invariant. The goal in this framework is to be able to produce an estimate of the system output, given a new set of inputs. If the output is a linear combination of inputs and past outputs, then the system is linear and is characterized by its impulse response. The impulse response is the output of the system when the input consists of a single sample equal to one, followed by zeros. Nonlinear systems cannot be characterized by their impulse response, although the response to an impulse input can used for comparison of different estimators.

Linear parametric approach: There are many different linear parametric models that are used to do system identification but the general theme between them all is that they assume a structure which can be expressed in terms of previous output samples, previous input signals and the noise term, $\epsilon(t)$, of the form:

$$y(t) = -a_1 y(t-1) \cdots - a_{q_a} y(t-q_a) + b_0 u(t) + b_1 u(t-1) + \cdots + b_{q_b} u(t-q_b) + \epsilon(t).$$

With a selected structure the coefficients are estimated using measured outputs and known inputs via maximum likelihood and regularized least-squares. These schemes are restricted to linear systems and have names like ARX and ARMAX and are highly used in the control theory literature [8][4][2].

To provide context for the kernel approach, let the linear system be represented by its impulse response, $h(t), t = 0, .., p-1$:

$$y(t) = h(0)u(t) + h(1)u(t-1) + \cdots + h(p-1)u(t-p+1) + \epsilon(t),$$

where p is a measure of the length of the impulse response. The impulse response may in fact have infinite length, but the order of this model, p, captures the bulk of the energy of the signal that is significant compared to the noise. Given a length of input signal equal to $(n + p - 1)$, the linear system can be written in matrix form as

$$\mathbf{Xh} = \mathbf{y}$$

where $\mathbf{X}$ is an $n \times p$ matrix in which $\mathbf{X}(i,:) = [u(t), \ldots, u(t-p+1)], i = 1 \ldots n$. The vector $\mathbf{h}$ is the impulse response, and $\mathbf{y}$ is the vector of outputs. With this representation, the parametric system identification becomes:

$$\hat{\mathbf{h}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$$

where λ is a regularization parameter. Given a new system input that is arranged into a $m \times p$ matrix $\tilde{\mathbf{X}}$, the estimated output is:

$$\hat{\mathbf{y}} = \tilde{\mathbf{X}}\hat{\mathbf{h}}.$$

Kernel Approach: In the kernelized version of the problem, an $n \times 1$ vector of coefficients is computed as

$$\mathbf{v} = (\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{y}$$

where $\mathbf{K}$ is an $n \times n$ kernel matrix that depends on the training data, $\mathbf{X}$, as well as hyper-parameters that control properties of the kernel such as the assumed decay of the impulse response. Note that now the matrix $\mathbf{X}$ is of size $n \times d$, where d is the maximum assumed history of the signal. It does not correspond to model order since the decay properties of the kernel take care of that, but d should be made large enough to accommodate the longest possible impulse response. Given a new system input the estimated output is:

$$\hat{\mathbf{y}} = \tilde{\mathbf{K}}\mathbf{v}.$$

where $\tilde{\mathbf{K}}$ depends on $\tilde{\mathbf{X}}$, $\mathbf{X}$, and the kernel hyper-parameters.
The regularization parameter λ, and the kernel parameters, are hyper-parameters that are determined using empirical Bayes. For example, if α is a kernel parameter, the values of α and λ are found which minimize the quantity:

$$\mathbf{y}^T(\mathbf{K}(\alpha) + \lambda\mathbf{I})^{-1}\mathbf{y} + \log\det(\mathbf{K}(\alpha) + \lambda\mathbf{I}).$$

This is done by simple search over or a range of possible hyper-parameter values.

Stable Spline Kernel: It is pointed out in [5, 7] that in using kernels for system identification, it is important to distinguish between the kernel as a measure of similarity between output signal vectors, and the measure of similarity of samples of the impulse response. For example, a common kernel in machine learning is a radial kernel. This is constant along the diagonal and decays away from the diagonal, corresponding to idea that inputs that are close to each other produce similar outputs. However, this does not capture the stability requirement of the impulse response for system identification, which is that the impulse response must decay with time. Otherwise, for example, a constant input would cause the output to blow up. To achieve this, a kernel for the impulse response is formulated that satisifes the stability properties, and this is in turn used to form the actual kernel used in the solving for the system outputs given the inputs.

The Stable Spline Kernel is first introduced in [7] and has the form:

$$\mathbf{K}_{SS} = \tilde{\mathbf{X}}\mathbf{S}\mathbf{X}^T \text{ where } \mathbf{S} \text{ is:}$$

$$\mathbf{S}_{ij} = \alpha^{\max(i,j)} \text{ for } \alpha \in [0, 1]$$

Note that the inner of the impulse response, $\mathbf{S}$, decays along the diagonal as well as away from it. This inner kernel of the impulse response is convolved with the input data to form the overall kernel, $\mathbf{K}_{SS}$.

This kernel is able to describe linear systems but does not do a good job when the system is nonlinear. The parameter α is determined using empirical Bayes and is the analog to determining the system order. The kernel is chosen to be part of the Reproducing Kernel Hilbert Space which says that this kernel follows the representer theorem and belongs in the Hilbert Space. For our purposes this just means it is able to represent the set of functions that we are interested in.

Nonlinear Stable Spline Kernel: The nonlinear stable spline kernel is described in [5] and builds off the stable spline kernel. The formulation of the nonlinear stable spline kernel follows from the stable spline kernel as follows:

$$\begin{aligned}\mathbf{z} =& (\tilde{\mathbf{X}}_i - \mathbf{X}_j)^T \\ \mathbf{K}_g =& \exp\left(-\frac{1}{\eta}\mathbf{z}^T\mathbf{S}\mathbf{z}\right) \\ \mathbf{K}_{NLSS} =& \mathbf{K}_{SS} \times \mathbf{K}_g\end{aligned}$$

where **S** is the same as for the stable spline kernel, and the last line is an element-wise product. This nonlinear stable spline kernel incorporates the decay of the stable spline kernel, and multiplies it with a Gaussian kernel that is a function of differences between inputs at different times. The effective basis of Gaussian kernel contains cross terms and higher powers of inputs, and also decays with time. The parameter η controls the extent of the nonlinear input interaction, and is also found through empirical Bayes.

3 Simulation Experiment Description

There are three ways in which we will explore the Nonlinear Stable Spline Kernel. A linear example, a nonlinear example and a nonlinear example with cross terms. We compare the results from the Nonlinear Stable Spline kernel (NLSS) to that of the Stable Spline (SS). The idea is to use the results of SS kernel as a basis for comparison. The three equations that are used to simulate the model are shown below:

$$y(t) = 0.5y(t-1) + 0.6u(t-2) - 0.2u(t-3) \tag{1}$$

$$y(t) = 0.5y(t-1) - 0.3y(t-2) - 0.5u(t-1) + 0.5u(t-2) + 0.5u(t-2)^2 \tag{2}$$

$$y(t) = -0.35y(t-1) - 0.8u(t-1) - 0.8u(t-2)^2 + 1.5\left((u(t-1))^3(u(t-3))^{1.8}\right) \tag{3}$$

These examples are discrete difference equations used to evaluate the performance of the kernel, these do not represent any real phenomenon but are there to illustrate the kinds of system that can be described in this framework.

For each example, training is done by applying a Gaussian random input signal into the system of length n. The output y is then corrupted with noise where σ is the strength of the noise. We then test the result by applying a periodic input signal $u(t) = 0.5\sin(5t)$ as our test signal. This is done to totally decouple the training signals from the test signals to really evaluate how well the kernel captures the properties of the unknown system. In order to get a distribution of results from this method we do a Monte-Carlo simulation of k runs. This also gives us an average of the performance of the kernel with respect to the test data. The error is measured as a relative mean squared error:

$$rel_{err} = \frac{\sqrt{mean((y_{test} - \hat{y})^2)}}{\sqrt{mean(y_{test}^2)}}$$

Figure 1 illustrates an example of what the training input signal looks like with its associated output along with the true impulse response that corresponds to the system.

4 Experiment Results

As a baseline, examples (1), (2) and (3) are tested with a noise strength $\sigma = 0.1$. The results are tabulated in Table 1 . Figures 2, 3 and 4 show the Monte-Carlo simulation results after 70 trial runs to give an idea of the kind of spread one can expect for capturing the impulse response for (1), (2) and (3) respectively. Each of the figures also shows a corresponding snap shot trial run of the true output y_{train}, the estimated output $\hat{y}$ and the input u which produced the results y.

From Figures 2, 3 and 4 we see that the NLSS kernel performs about as well as the SS kernel for the linear case with moderate amount of noise and that the NLSS can capture the impulse response for

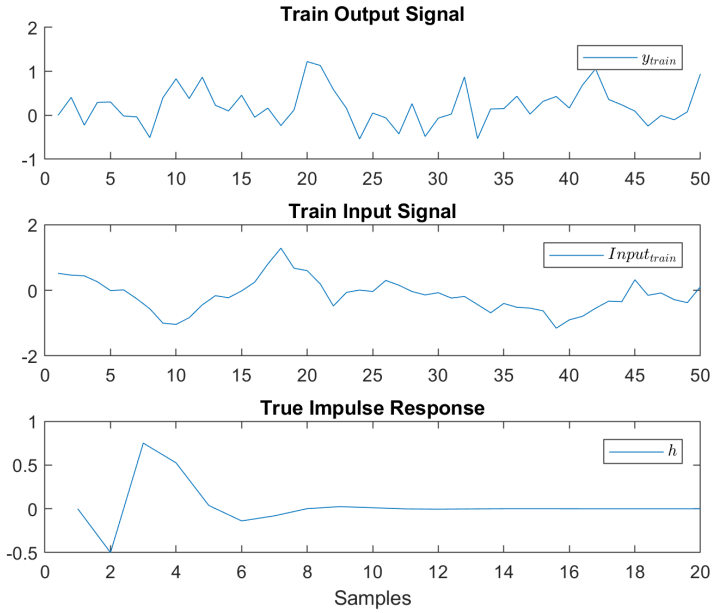

Figure 1: Snap shot of training of the nonlinear example (2) with $\sigma = 0.3$.

Table 1: Comparison Results between Stable Spline and Nonlinear Stable Spline

Exp	$SS_{rel_{err}}$	$NLSS_{erel_{err}}$	Noise σ
Linear Example (1)	0.0825	0.1012	0.1
NonLinear Example (2)	1.1348	0.7089	0.1
NonLinear Example (3)	0.9082	0.5614	0.1

the nonlinear cases whereas the SS kernel does not.

We further investigate the effect of noise on the NLSS kernel and compare it against the SS kernel. The results are summarized in Table 2 while the analysis is further broken down into each case.

4.1 The Linear Case (1)

What is observed in Table 2 is that for a strong noise strength $\sigma = 1$ the error for the NLSS kernel is almost twice as bad as that for the SS kernel. Figure 5 and 6 show that the SS kernel although not perfect is still able to reproduce a reasonable signal, while the NLSS kernel is doing a bad job. This seems to suggests that the SS is well equiped at noise rejection and the NLSS is overcomplicating the system in the presense of the noise. Figure 7 shows the Monte-Carlo simulations at a noise strength

Table 2: Comparison Results between Stable Spline and Nonlinear Stable Spline with increasing noise

	Linear		NonLinear Simple		NonLinear Complex	
Noise σ	$SS_{rel_{err}}$	$NLSS_{rel_{err}}$	$SS_{rel_{err}}$	$NLSS_{rel_{err}}$	$SS_{rel_{err}}$	$NLSS_{rel_{err}}$
0.01	0.0089	0.0096	1.1874	0.4509	0.8512	0.3518
0.1	0.0744	0.0872	1.1424	0.8187	0.7509	0.4426
0.3	0.1822	0.2834	1.1225	1.1196	0.7839	0.6439
0.5	0.3314	0.4440	1.4192	1.7013	0.9134	0.8540
0.7	0.4438	0.8869	1.5799	1.7971	1.0101	1.2260
1.0	0.6323	1.0633	1.8683	2.7164	1.106	1.4012

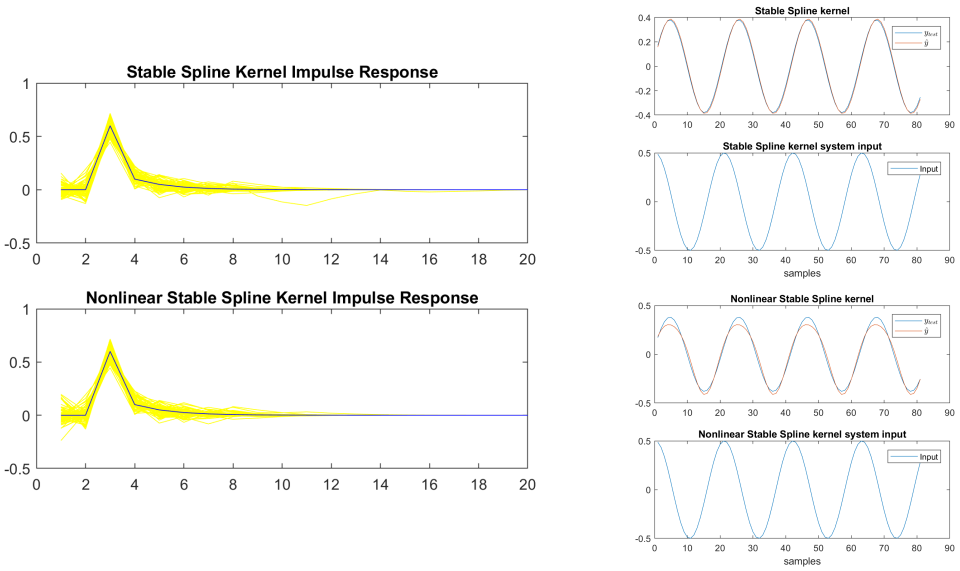

Figure 2: Monte-Carlo simulation of the impulse response to the linear example (1) with $\sigma = 0.1$ and a snap shot of the input/output estimate for both the SS and NLSS kernels.

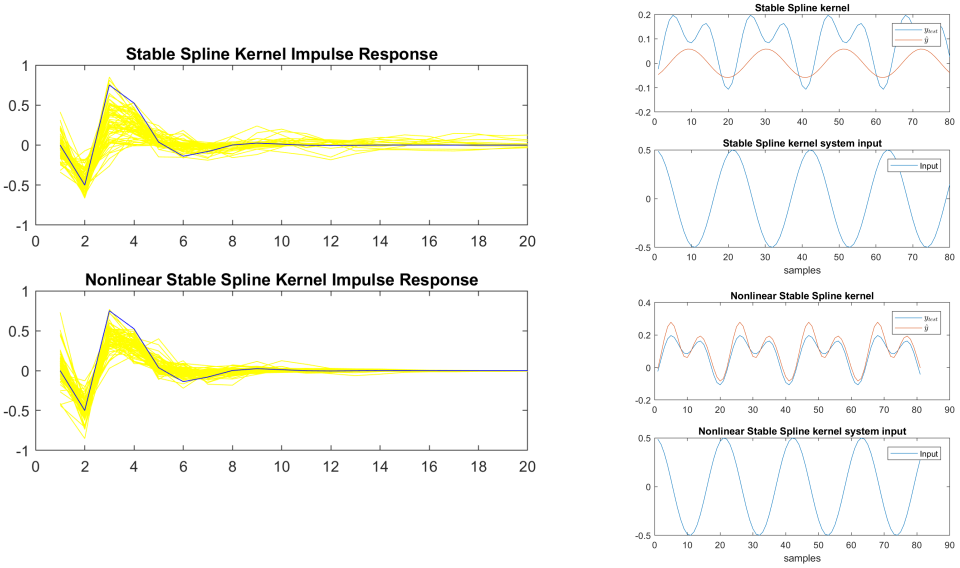

Figure 3: Monte-Carlo simulation of the impulse response to the nonlinear example (2) with $\sigma = 0.1$ and a snap shot of the input/output estimate for both the SS and NLSS kernels.

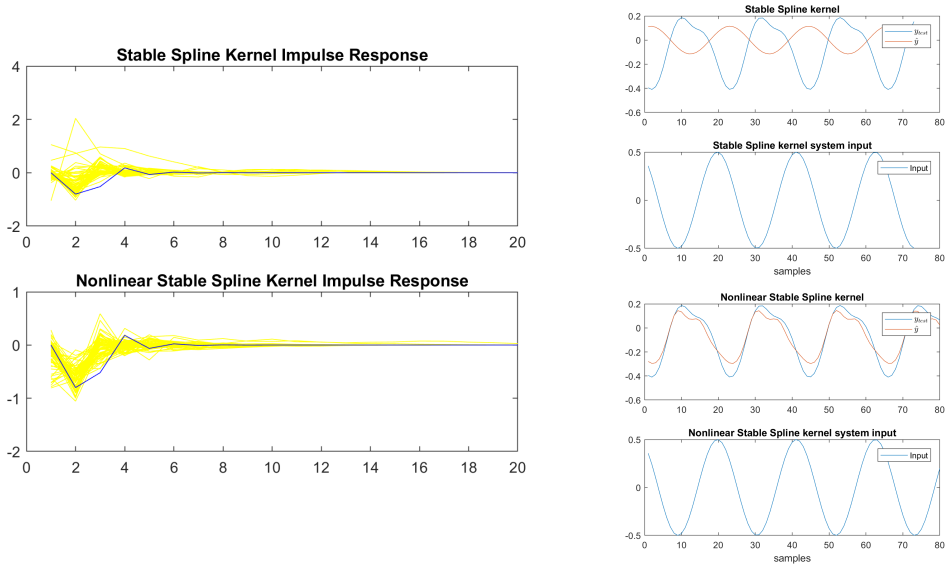

Figure 4: Monte-Carlo simulation of the impulse response to the nonlinear example (3) with $\sigma = 0.1$ and a snap shot of the input/output estimate for both the SS and NLSS kernels.

of $\sigma = 1$ to illustrate the variability observed in the estimates impulse response, the Figure does illustrate that with high noise the variability in the results are quite high.

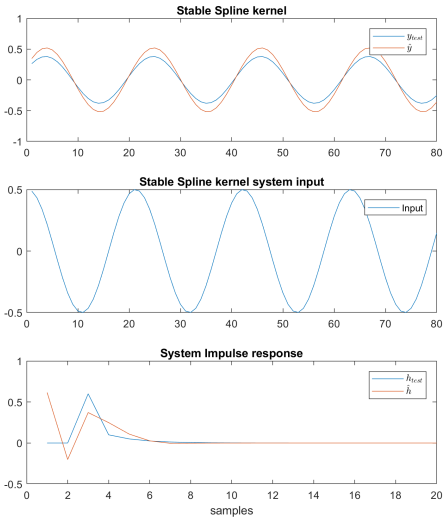

Figure 5: Example trial run simulation for the SS kernel of the output y_{test}, $\hat{y}$ and the input u to the linear example (1) with $\sigma = 1$.

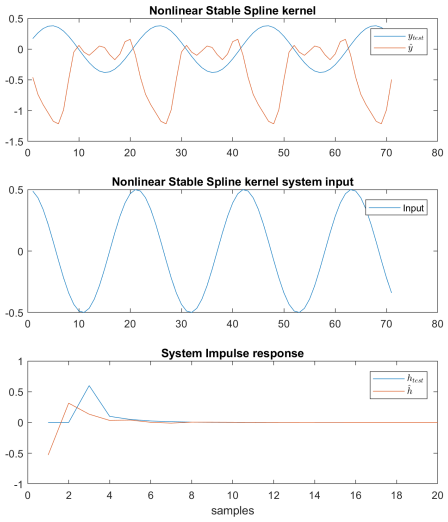

Figure 6: Example trial run simulation for the NLSS kernel of the output y_{test}, $\hat{y}$ and the input u to the linear example (1) with $\sigma = 1$.

4.2 The Nonlinear Case (2)

What is curious is that the error for both the SS and NLSS for some threshold of noise strength σ start to look about the same. In the nonlinear example (2) this appears to occur at around $\sigma = 0.3$. Taking a closer look at what the signals look like is shown in Figure 9 and 8. What is observed between the two figures is that even though the errors are approximately the same in magnitude, the signals are produc-

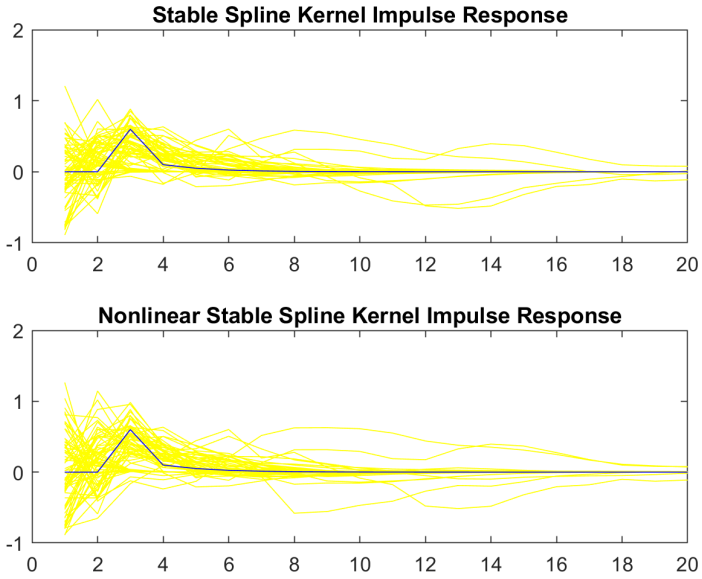

Figure 7: Monte-Carlo simulation of the impulse response to the linear example (1) with $\sigma = 1.0$.

ing the error in different ways. The general shape of the signal from the NLSS kernel appears correct but the scaling is wrong, where as the SS kernel has the right scale but does not capture the shape at all.

Figures 11 and 10 show a snap shot of what seems to happen as the noise strength is increased to $\sigma = 0.7$. The NLSS is producing results that look very similar to that of the SS. This seems to imply that as the noise increases for a nonlinear system the NLSS kernel starts to perform about as well as the SS kernel.

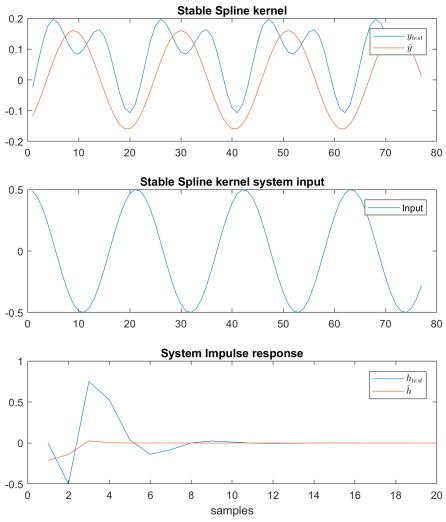

Figure 8: Example trial run simulation for the SS kernel of the output y_{test}, $\hat{y}$ and the input u to the nonlinear example (2) with $\sigma = 0.3$.

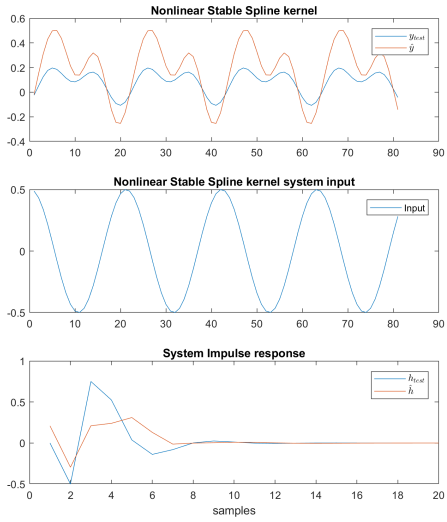

Figure 9: Example trial run simulation for the NLSS kernel of the output y_{test}, $\hat{y}$ and the input u to the nonlinear example (2) with $\sigma = 0.3$.

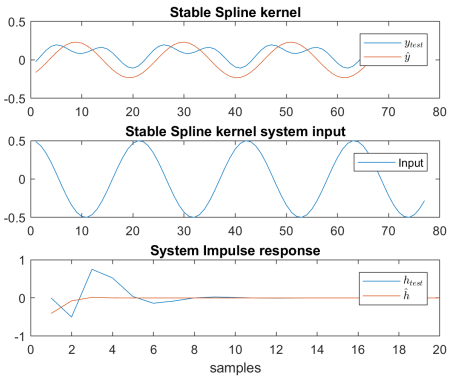

Figure 10: Example trial run simulation for the SS kernel of the output y_{test}, $\hat{y}$ and the input u to the nonlinear example (2) with $\sigma = 0.7$.

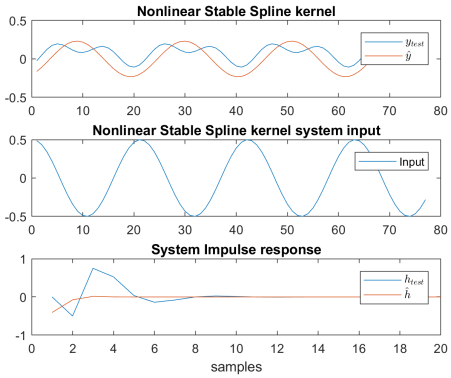

Figure 11: Example trial run simulation for the NLSS kernel of the output y_{test}, $\hat{y}$ and the input u to the nonlinear example (2) with $\sigma = 0.7$, showing signs of exhibiting similar behaviour to the SS kernel.

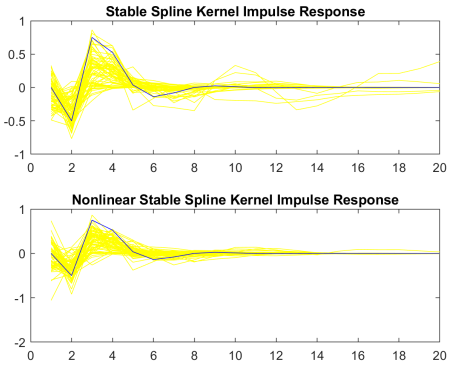

Figure 12: Monte-Carlo simulation of the impulse response to the nonlinear example (2) with $\sigma = 0.3$.

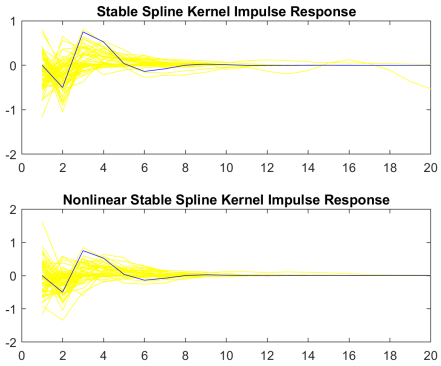

Figure 13: Monte-Carlo simulation of the impulse response to the nonlinear example (2) with $\sigma = 0.7$.

4.3 The Nonlinear Case (3) with cross terms

As the complexity of the nonlinearity is increased as is the case in example (3) with cross terms the threshold where the SS kernel performs to the same level of the NLSS kernel increased for this particular example. Here the noise threshold appears to be around $\sigma = 0.5$. The individual trial runs start to look similar in nature to what was uncovered in example (2), that is that the NLSS kernel starts to look like the SS kernel estimate. Figure 14 shows that at the noise strength of $\sigma = 0.5$ that the estimated impulse responses start looking similar.

5 Conclusion and Future Work

The conclusion from these tests show that if the system is infact linear that the SS kernel is the better kernel to use, however for low noise the results between the SS and NLSS are comparable. If the system contains some nonlinearity to it the NLSS kernel works very well as long as the noise is not too strong. There seems to exist a noise strength σ for a given system where both the NLSS and the SS produce errors in the estimation of the impulse response that are comparable. The SS kernel seems to be able to consistenly produce estimates that capture the overall shape of the response even in the presense of a lot of noise. The NLSS is better suited at capturing details and nonlinearities

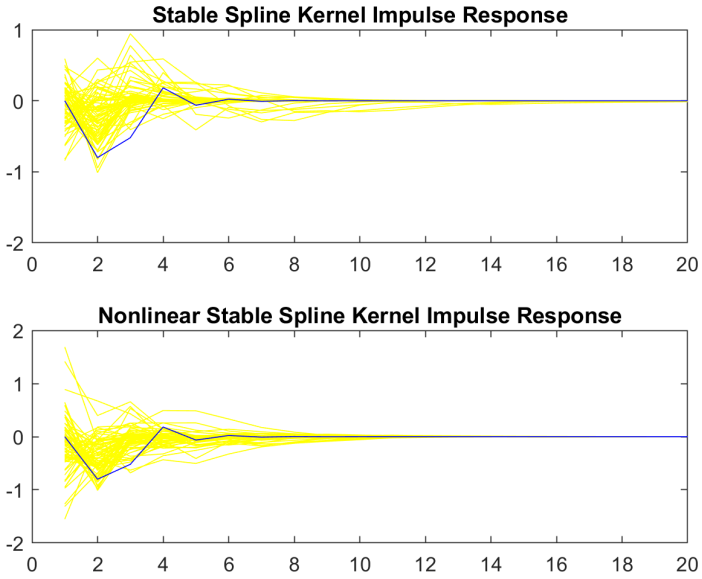

Figure 14: Monte-Carlo simulation of the impulse response to the nonlinear example (3) with $\sigma = 0.5$.

when the noise strength is not too high. A more complex nonlinearity system might help the NLSS be better at noise rejection, but such claims would need to be confirmed with more nonlinear examples. The examples used to evaluate the method are not from real physical realizations and so the next step would to find real physical examples to better evaluate the abilities of the NLSS under real world examples.

Aspects that have not been covered in this work is how well these methods are at capturing time delay in a system. This analysis is also all done offline, one of the nice advantages to the parametric approach is that the estimation of the system can be done online and recursively. This has the advantage of being able to track gradual changes to the system overtime. Very recent research seems to be expanding on the general idea of the SS and NLSS kernel to achieve this kind of flexibility.[9]

As of right now even though the kernels are able to capture the behaviour of linear and nonlinear systems it would be great to find ways to tie in traditional control theory design methodologies to work in the kernelized framework. It is not clear with a kernelized blackbox how to design a controller to send control signals into the kernel model to produce desired output trajectories.

References

[1] A. Chiuso G. Pillonetto, M. H. Quang. A new kernel-based approach for nonlinear system identification. *IEEE Transactions on Automatic Control*, 56(12):2825–2840, 2011.

[2] Lennart Ljung. *System Identification*, pages 163–173. Birkhäuser Boston, Boston, MA, 1998.

[3] Lennart Ljung. Perspectives on system identification. *Annual Reviews in Control*, 34(1):1 – 12, 2010.

[4] Lennart Ljung, Håkan Hjalmarsson, Henrik Ohlsson, Reglerteknik, Skolan för elektro-och systemteknik (EES), and KTH. Four encounters with system identification. *European Journal of Control*, 17(5-6):449–471, 2011.

[5] Gianluigi Pillonetto. The interplay between system identification and machine learning. 2016. arXiv:1612.09158.

[6] Gianluigi Pillonetto, Francesco Dinuzzo, Tianshi Chen, Giuseppe De Nicolao, and Lennart Ljung. Kernel methods in system identification, machine learning and function estimation: A survey. *Automatica*, 50(3):657 – 682, 2014.

[7] Gianluigi Pillonetto and Giuseppe De Nicolao. A new kernel-based approach for linear system identification. *Automatica*, 46(1):81 – 93, 2010.

[8] Sofia Rachad, Benayad Nsiri, and Bahloul Bensassi. System identification of inventory system using arx and armax models. 8:2 83 – 2 94, 12 2015.

[9] José Daniel A. Santos and Guilherme A. Barreto. An outlier-robust kernel rls algorithm for nonlinear system identification. *Nonlinear Dynamics*, 90(3):1707–1726, Nov 2017.