ECE 5390 Practicum Assignment 1

Modeling from Experimental Data

Griffin Davis

January 27, 2022

1 Introduction

Practicum 1 explores the modeling of first and second order systems given the data points of the system's step response. System modeling is completed and plotted using MATLAB.

The rest of this report is organized as follows. Section 2 covers the search method for parametric identification of the system. Section 3 discusses the results of the search in reference to a first order system. Section 4 discusses the results of the search in reference to a second order system. Section 5 discusses the practicum as a whole and draws conclusions.

2 Parametric Identification Through N-Search

Practicum 1 calls for the approximation of a first and second order system through parametric identification of terms *k*, α, and β. A brute force method using N-search is used to determine the parameters yielding the minimum Least Squares Error.

Before searching, an estimate of α and *k* for the first order system is determined using equations:

$$\tau = \frac{1}{\alpha}$$
$$\frac{k}{\alpha} = f(t)_{steady\_state}$$

The search involves iterating a set number of times, tweaking the magnitude by which parameters *k* and α—or β in the second order—are changed and either increasing or decreasing values by that magnitude in accordance with the operation yielding the lower least squares error.

MATLAB code of the search methods for the first and second order systems can be found in **Appendix A** and **Appendix B**.

3 First Order System Modeling

The N-Search method discussed in section 2 assuming a first order system yields a transfer function with parameters:

$$k = 0.29758$$
$$\alpha = 0.29019$$

The above values result in a least squares error of 0.101436. Further tests with a greater number of iterations and more precise alterations in magnitude determined this to be the lowest obtainable least squares error within a reasonable search time.

The experimental model can be seen plotted against the provided data in **Figure 1**.

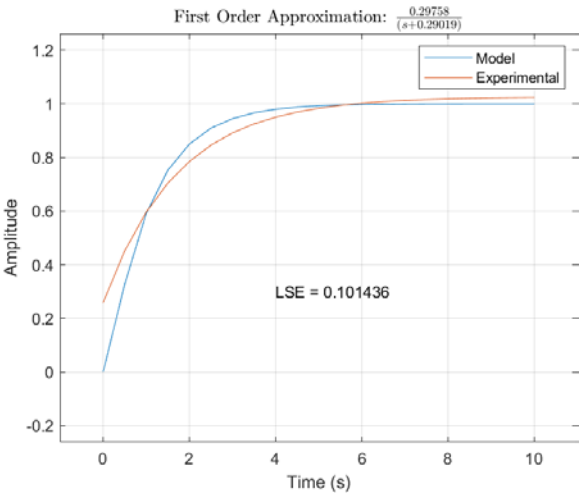

Fig. 1: First Order Approximation of Step Response

4 Second Order System Modeling

Applying the search method to a second order assumption first requires an assumption of β. β is first tested as equal to α, but this assumption results in an

exponential step response rather than a critically damped, logarithmic response. α is doubled and the equivalence tested again for a least squares error of 0.0636809. The parameters are identified as:

$$k = 0.29758$$
$$\alpha = 0.51938$$
$$\beta = 0.58038$$

Further tests were explored and again the least squares error of ~0.064 was the lowest obtainable error within a reasonable search time.

The experimental model can be seen plotted against the provided data in **Figure 2**.

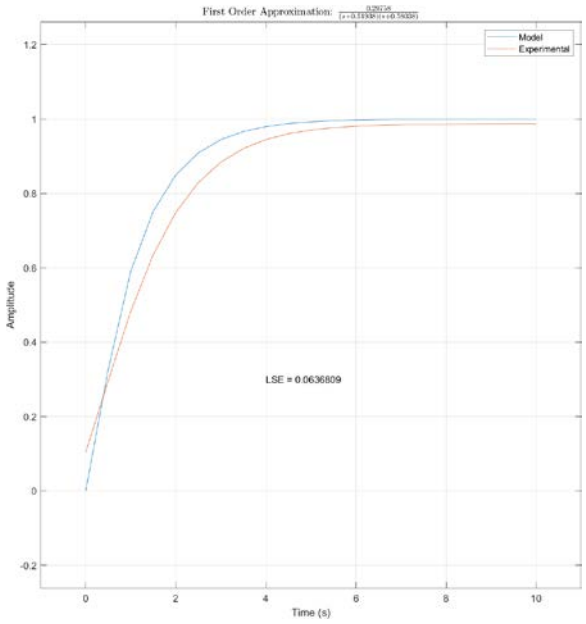

Fig. 2: Second Order Approximation of Step Response

5 Discussion and Conclusion

Due to the lower least squares error of 0.0636809 found in the second order approximation of the step response, the system is more likely to be a second order system than a first order system. It can also be seen in **Figures 1 & 2** that the second order system reaches steady state at close to the same period of 4τ than the first order system. The second order system comes closer to an initial start of 0 as well and thus more accurately represents a system step response.

Practicum 1 is a great introduction to system modeling and parametric identification through the relatively complex algorithm required to accurately calculate the parameters through N-search.

Appendices

Appendix (A): First Order Search

```
for i=1:1000
    if LSE == 0
        break
    end

    if mod(i,2)
        % work on alpha
        alpha_pos = alpha + aMag_pos;
        alpha_neg = alpha - aMag_neg;

        aLSE_pos = getLSE(alpha_pos, k);
        aLSE_neg = getLSE(alpha_neg, k);

        if aLSE_pos > prevLSE && aLSE_neg > prevLSE
            aMag_pos = aMag_pos - mag_order;
            aMag_neg = aMag_neg - mag_order;
        else
            if aLSE_pos > aLSE_neg
                aMag_pos = aMag_pos - mag_order;
                alpha = alpha_neg;
            else
                aMag_neg = aMag_neg - mag_order;
                alpha = alpha_pos;
            end
        end
    else
        % work on k
        k_pos = k + kMag_pos;
        k_neg = k - kMag_neg;

        kLSE_pos = getLSE(alpha, k_pos);
        kLSE_neg = getLSE(alpha, k_neg);

        if kLSE_pos > prevLSE && kLSE_neg > prevLSE
            kMag_pos = kMag_pos - mag_order;
            kMag_neg = kMag_neg - mag_order;
        else
            if kLSE_pos > kLSE_neg
                kMag_pos = kMag_pos - mag_order;
                k = k_neg;
            else
                kMag_neg = kMag_neg - mag_order;
                k = k_pos;
            end
        end
    end
    prevLSE = LSE;
    LSE = getLSE(alpha, k);
end
```

Appendix (B): Second Order Search

```
for i=1:1000
    if LSE == 0
        break
    end
    m = mod(i,3);
    if m==2
        % work on alpha
        alpha_pos = alpha + aMag_pos;
        alpha_neg = alpha - aMag_neg;

        aLSE_pos = getLSE(alpha_pos, k);
        aLSE_neg = getLSE(alpha_neg, k);

        if aLSE_pos > prevLSE && aLSE_neg > prevLSE
            aMag_pos = aMag_pos - mag_order;
            aMag_neg = aMag_neg - mag_order;
        else
            if aLSE_pos > aLSE_neg
                aMag_pos = aMag_pos - mag_order;
                alpha = alpha_neg;
            else
                aMag_neg = aMag_neg - mag_order;
                alpha = alpha_pos;
            end
        end
    elseif m==1
        % work on k
        k_pos = k + kMag_pos;
        k_neg = k - kMag_neg;

        kLSE_pos = getLSE(alpha, k_pos);
        kLSE_neg = getLSE(alpha, k_neg);

        if kLSE_pos > prevLSE && kLSE_neg > prevLSE
            kMag_pos = kMag_pos - mag_order;
            kMag_neg = kMag_neg - mag_order;
        else
            if kLSE_pos > kLSE_neg
                kMag_pos = kMag_pos - mag_order;
                k = k_neg;
            else
                kMag_neg = kMag_neg - mag_order;
                k = k_pos;
            end
        end
    else
        % work on beta
        beta_pos = beta + bMag_pos;
        beta_neg = beta - bMag_neg;

        bLSE_pos = get2LSE(alpha, beta_pos, k);
        bLSE_neg = get2LSE(alpha, beta_neg, k);

        if bLSE_pos > prevLSE && bLSE_neg > prevLSE
            bMag_pos = bMag_pos - mag_order;
            bMag_neg = bMag_neg - mag_order;
        else
            if bLSE_pos > bLSE_neg
                bMag_pos = bMag_pos - mag_order;
                b = beta_neg;
            else
                bMag_neg = bMag_neg - mag_order;
                b = beta_pos;
            end
        end
    end
    prevLSE = LSE;
    LSE = get2LSE(alpha, beta, k);
end
```