Computing System Congestion Management Using Exponential Smoothing Forecasting

James F Brady
Capacity Planner for State of Nevada
575 E. Third Street
Carson City, Nevada 89701
(001) (775) 684-5891
jfb Brady@admin.nv.gov

ABSTRACT

An overloaded computer must finish what it starts and not start what will fail or hang. A congestion management algorithm, the author developed, effectively manages traffic overload with its unique formulation of Exponential Smoothing forecasting. This set of equations resolve forecasting startup issues that have limited the model's adoption as a discrete time series predictor. These expressions also satisfy implementation requirements to perform calculations using integer math and be able to reset the forecast seamlessly. A computer program, written in C language, which exercises the methodology, is downloadable from [GitHub](#).

CCS Concepts

• General and referen→Cross-computing tools and techniques
→Performance

Keywords

congestion management; overload control; exponential smoothing

1. INTRODUCTION

There are countless examples of computing systems crashing or slowing to a virtual standstill due to an offered workload that exceeds available capacity. Application developers usually ignore the problem until disaster strikes and they are forced to address the issue. Strategies vary but are typically focused on one of three restrictive actions, capping queue length, throttling CPU usage, or limiting response time. When the chosen threshold is reached, action is taken to deny or delay new requests for service but permit transaction sequences underway to continue.

The approach taken here, patented for telecom products by a major manufacturer with this author as the sole inventor [US7301903B2](#), is based on limiting response time using measurements that are statistically smoothed with a uniquely formulated Exponential Smoothing forecasting model. Exponential Smoothing has not gained a great deal of acceptance by practitioners because of some implementation issues thought to be insurmountable which are addressed within this methodology.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

ICCD 2020, March 9–12, 2020, Silicon Valley, CA, USA

© 2020 Association for Computing Machinery.

ACM ISBN 978-1-4503-7644-0/20/03...\$15.00

<https://doi.org/10.1145/3388142.3388146>

Section 2 sets the stage with a web-based environment where overload control is implemented using exponentially smoothed response time measurements. Section 3 provides an introductory overview of the Exponential Smoothing forecasting model and compares its statistical properties to a more commonly used method. Issues associated with Exponential Smoothing as a prediction tool are discussed in Section 4 and adaptation of the complete overload control algorithm is described in Section 5. An overview of the computer program which exercises the algorithm is contained in Section 6, and some summary remarks are made in Section 7. The mathematics used throughout this discussion is time series based and the key reference, [2], can be purchased [online](#). As a convenience, links are setup in this document for [internal](#) as well as [external](#) references and [section](#) / [figure](#) locations. To return from internal links use the Alt + left arrow.

2. COMPUTING ILLUSTRATION

Figure 1 is an example of a computing system congestion management setup that applies response time measurements. It shows a Web/App server and a Database server where round trip time is captured by the Web/App Server for a selected set of HTTP GET and POST events. These latency numbers are used for congestion control by statistically smoothing them to produce an “expected” response time which is compared to an overload threshold value. If the threshold is exceeded, the next request for service is denied or delayed, but transaction sequences already in progress continue unabated to completion.

This is only one example of how smoothed response time can be used for congestion control. The implementation contained in US patent [US7301903B2](#) is traffic flow control for a multimedia telecommunications system.

Figure 1. Congestion Management Illustration

The focus of this discussion is not a specific congestion management implementation, but the algorithm used to produce the smoothed response times needed for its operation. Exponential Smoothing time series forecasting is chosen as the basis of the

smoothing algorithm because it requires a minimum amount of data to be retained and weighs recent data more heavily than older data. The next section contains an overview of Exponential Smoothing and compares its statistical properties to those of the more commonly used Moving Average.

3. EXP SMOOTH Vs MOVING AVERAGE

Exponential Smoothing is a statistical forecasting technique that provides a time series forecast using the most recent data value and the previous forecast as input. Named for its data discounting properties, Exponential Smoothing lessens the weight given to a specific data point geometrically with the addition of subsequent observations to the time series. Although very small after many observations have occurred, the weight a data value contributes to the forecast is never zero. Eq. 1 is the most basic form of an exponentially smoothed forecast called Single Exponential Smoothing.

$$F_t = S_t(x) = \alpha x_t + (1 - \alpha)S_{t-1}(x) \quad (1)$$

Where:

F_t = forecast for point in time t

$S_t(x)$ = first smoothed statistic at point in time t

α = smoothing constant $0 < \alpha < 1$

x_t = observation for point in time t

Weight Given $S_i(x) = \alpha, \alpha(1 - \alpha), \dots, \alpha(1 - \alpha)^{i-1} \dots$

This approach is contrasted with a Moving Average, Eq. 2, which produces its forecast by computing the arithmetic mean of the most recent N data values. This method weighs all N values in the calculation range equally and gives no weight to data outside that range.

$$M_t = \frac{x_t + x_{t-1} + \dots + x_{t-N+1}}{N} \quad (2)$$

Where:

M_t = moving average forecast

N = number of moving average data points

Weight Given $M_i = \frac{1}{N}, \frac{1}{N}, \dots, \frac{1}{N} \dots$

A Moving Average requires N data points to be stored for initiating and continuing the forecast while Exponential Smoothing needs only the most recent data point and the previous forecast. Figure 2 shows the difference in weight given to data for Exponential Smoothing Vs Moving Average. Each observation in the twenty point Moving Average gets an equal weight of $\frac{1}{N} = \frac{1}{20} = .05$, whereas, the most recent observation being Exponentially Smoothed gets a weight of $\alpha = .10$ and the oldest observation gets $\alpha(1 - \alpha)^{20-1} = .10(1 - .10)^{19} = .013509$.

Figure 2. Weight Given Time Series Samples

For details concerning the mathematical formulation of both models and a discussion of their data discounting characteristics see [2].

4. EXP SMOOTHING ISSUES

Many practitioners have found Exponential Smoothing to be an intuitively appealing approach to forecasting but were disappointed with the outcome. This author believes the dissatisfaction results from a less than adequate approach to two basic implementation issues.

1. Forecasting startup
2. Forecasting a ramp

The first issue results from the fact that significant weight is given to the initial forecast in Eq. 1, $S_0(x)$, for several subsequent forecasting periods. For example, the sum of the twenty weights in the Figure 2 blue column is .878423, leaving .121577 for the initial forecast, which is more than the .100000 given to the most recent data point. This problem is mitigated by restructuring Eq. 1 and applying that restructured formula at startup.

The second issue is important because Eq. 1 forecasts a ramp with a bias and response times ramp up rapidly as a system approaches a congested state.

4.1 Forecasting Startup

The significant weight given the initial forecast is demonstrated by expanding Eq. 1 over time as shown in Eq. 3 and detailed in Appendix A.

$$S_t(x) = \alpha x_t + \alpha(1 - \alpha)x_{t-1} + \dots + (1 - \alpha)^i S_0(x) \quad (3)$$

The Eq. 3 expansion illustrates that after i observations the weight given the initial estimate, $S_0(x)$, is $(1 - \alpha)^i$, which implies that when $\alpha = .10$ and $i = 10$ the weight given $S_0(x) = .348678$. This tenth forecast weight is over three times the .100000 weight given the most recent observation, x_t , and even greater for forecasts $0 < i < 10$.

Figure 3 extends this example to twenty points in time where the weight given $S_0(x)$ is compared to the weight assigned to the sum of the observations, $\sum S_i(x)$. The weight attributed to $\sum S_i(x)$ does not exceed that for $S_0(x)$ until the seventh observation, $i = 7$, where $\sum S_i(x) = .521703$ and $S_0(x) = .478297$.

$$\text{Weight Given } \sum S_i(x) = \alpha + \alpha(1 - \alpha) + \dots + \alpha(1 - \alpha)^{i-1}$$

$$\text{Weight Given } S_0(x) = (1 - \alpha), (1 - \alpha)^2, \dots, (1 - \alpha)^i$$

$$b_t = \frac{\alpha}{(1-\alpha)} [S_t(x) - S_t^{[2]}(x)] \quad (9)$$

A derivation of these expressions is contained in [2]. Higher order terms, such as a quadratic component, can be added but they tend to increase volatility in the forecast.

4.3 Algorithm Summary

Eq. 4 addresses issue 1 by eliminating the dominant weight given to the initial forecast and assigning equal weight to the first n observations incrementally. Eq. 6 through Eq. 9 resolve issue 2 by implementing Double Exponential Smoothing on an ongoing basis, thus tracking a ramp without a bias.

Eq. 4 is used at startup to produce the response time forecast when $n \leq \lfloor \frac{1}{\alpha} \rfloor$ and the Double Exponential Smoothing equations are applied when generating ongoing forecasts. Eq. 10 is implemented at Double Exponential Smoothing startup, $n = \lfloor \frac{1}{\alpha} \rfloor$, making $S_t(x)$ the forecast.

$$S_{t=n=\lfloor \frac{1}{\alpha} \rfloor}^{[2]}(x) = S_{t=n=\lfloor \frac{1}{\alpha} \rfloor}(x) \quad (10)$$

The only data items stored ongoing in this Double Exponential Smoothing setup are n , $S_t(x)$, and $S_t^{[2]}(x)$.

5. ALGORITHM IMPLEMENTATION

There are two implementation issues to consider when adapting the formulas just discussed to a real time forecasting environment like the one shown in Figure 1. First, there is a potential requirement to create an integer math formulation of the expressions and second, a mechanism needs to be in place to reset the algorithm based on the time that has elapsed since the last update.

An integer math formulation is likely necessary if the algorithm is implemented in the operating system as part of a device driver. Linux, as an example, does not reliably support floating point math in the kernel. The second issue focuses on resetting the forecasting algorithm if it has been a long time since an observation has taken place.

5.1 Integer Math Formulation

The integer math conversion is produced by substituting $\frac{1}{n}$ for α where $0 < \alpha \leq .5$ into the equations previously introduced and rearranging. The results of these manipulations are shown below with Eq. 12 as an integer replacement for Eq. 4 and Eq. 13 thru Eq. 19 integer replacements for Eq. 1 and Eq. 5 thru Eq. 10, ongoing.

The integer formulation halves the range of α because an $\alpha > .5$ makes $n_\alpha = 1$ and b_t in Eq. 16 undefined. Eq. 12 and Eq. 16 through Eq. 18 perform multiplication, addition, and subtraction before division based on standard C language operator precedence. The $\lfloor \text{integer floor} \rfloor$ bars are used sparingly below to avoid unnecessary clutter. Appendix B provides a detailed mapping of these integer equations to their non-integer counterparts.

For: $0 < \alpha \leq .5$

$$\text{Let:} \quad n_\alpha = \left\lfloor \frac{1}{\alpha} \right\rfloor \quad (11)$$

$$\text{Startup: } n \leq n_\alpha \quad F_t = S_t(x) = \frac{x_t + (n-1)S_{t-1}(x)}{n} \quad (12)$$

$$\text{Ongoing:} \quad F_t = a_t + b_t L \quad (13)$$

$$\text{Where:} \quad L = 1 \quad (14)$$

$$a_t = 2S_t(x) - S_t^{[2]}(x) \quad (15)$$

$$b_t = \frac{S_t(x) - S_t^{[2]}(x)}{(n_\alpha - 1)} \quad (16)$$

$$S_t(x) = \frac{x_t + (n_\alpha - 1)S_{t-1}(x)}{n_\alpha} \quad (17)$$

$$S_t^{[2]} = \frac{S_t(x) + (n_\alpha - 1)S_{t-1}^{[2]}(x)}{n_\alpha} \quad (18)$$

$$\text{Initial Condition:} \quad S_{t=n=n_\alpha}^{[2]}(x) = S_{t=n=n_\alpha}(x) \quad (19)$$

Although this integer formulation halves the range of the smoothing constant, there are few forecasting situations where over 50% of the forecasting weight should be given to the current observation. One way to minimize the truncation error incurred using integer data types is to choose a smoothing constant with an integer inverse, e.g., $n_{\alpha=.1} = \lfloor \frac{1}{.1} \rfloor = 10$ and $n_{\alpha=.2} = \lfloor \frac{1}{.2} \rfloor = 5$. The default setting in the [GitHub](#) computer program is $n_\alpha = 10$.

These equations are used to produce the results contained in Figure 6 for $n_\alpha = 10$ using the input data and C language computer program on [GitHub](#).

This is the same input data included in US patent [US7301903B2](#). The first 10 red forecast values are the arithmetic mean, Eq. 12, but after that, $t > 10$, the Double Exponential Smoothing portion of the algorithm, Eq. 13, takes control of the forecast. The integer setup is preferred by this author because it is fast, not resource intensive, and can be implemented at the operating system kernel level.

Figure 6. Smoothing Algorithm Integer Math Formulation

5.2 Forecast Reset

Because response times are event based, their sampling rate increases with higher traffic volume, allowing for quick reaction to an overload condition. Contrast this type of metric with an interval-based measurement like CPU busy which requires multiple fixed time interval samples to be taken that are a minimum of one second in duration. CPU busy also suffers from the fact that it is a resource busy indicator, not a direct measure of system congestion level.

Response times are a system congestion barometer only if the time between samples is small because, if it has been a long time since the last observation, e.g., five seconds, the smoothed response time is stale and the statistically smoothed forecast, F_t , needs to be reset. The most glaring example of this situation occurs in the laboratory when overload traffic is being offered to a system and is suddenly stopped and restarted below the overload level. The congestion management controls that are set will remain in effect for a few event sequences until the forecasting

algorithm reacts to the smaller response times being produced by the reduced traffic rate.

Figure 7. Smoothing Algorithm Integer Math with Reset

The solution to this problem is to reset the forecasting algorithm based on the time since the last update. This is accomplished by storing the wall clock time of the last update and if it has been at least, the example, five seconds since this last measurement, the current observation number is set to one, $n = 1$, from its ongoing value, n_{α} . This procedure resets the forecasting mechanism to the startup state, Eq. 12, making the current forecast equal to the value of the current observation, x_t . Since it has been a long time since the last event has occurred, this event's response time will be small or, if it is not, a system failure of some kind has likely taken place. The Figure 7 ramp model illustrates this mechanism when $n_{\alpha} = 5$ and the reset time delay begins at $t = 11$.

This reset example illustrates that during the averaging periods, $t = 1 - 5$ and $t = 12 - 16$, the **forecasts** are increasing at a much lower rate than the **observed** values but when Double Exponential Smoothing kicks in, $t = 6 - 10$ and $t = 17 - 25$, the forecast asymptotically approaches the observations. Of course, the forecast exactly matches the observation at $t = 12$ since it is the first event after a 6 second pause occurred with a 5 second reset interval.

6. COMPUTER PROGRAM

The equations and diagrams in the previous three sections explain the algorithm but may not be as helpful to the practitioner as the computer program on [GitHub](#). This C language program, which implements the equations in Section 5.1 and the forecast reset features of Section 5.2, can be downloaded, compiled, and run.

The repository documentation, time_series_smooth_doc.pdf, contains three parts:

1. Program Input
2. Program Output
3. Program Source Listing

To compile and run the program:

1. Download the time_series_smooth repository
2. Go to the src directory and execute compile_time_series_smooth.bat (installed gcc compiler is required)
3. The time_series_smooth.exe binary is created in the demo directory
4. Go to the demo directory and execute time_series_smooth.bat
5. The "Observe" and "Forecast" numbers listed in Figure

6 are produced

Program options:

1. -h = help
2. -n = n_{α} - integer value of $[1/\alpha]$ default is 10
3. -r = reset smoother at count value plus one
4. -t = reset smoother time interval default is 5 seconds
5. -w = write verbose output to comma delimited file

The function in the program that contains the full algorithm is "time_series_smooth()". This function's source code, call in the main program, prototype, and data structure (EXP_SMOOTH_DATA) are fully commented.

7. SUMMARY

Managing congestion in a computing system is a difficult challenge that is usually ignored until an overload crashes the system or leaves it in an indeterminant state. This paper describes an approach to the problem that was applied in a telecommunication setting and worked so well the corporation who developed the technique filed for and obtained U.S. patent rights. Section 2 illustrates how this methodology can be adapted to a computing system environment using response time measurements between a Web/App server and Database server. This computing system example provides implementation context for the primary focus of the discussion, the forecasting algorithm.

The algorithm applies Exponential Smoothing forecasting to an event driven time series of response time measurements, statistically smoothing them, and comparing the result to an overload threshold. If the threshold is exceeded, new requests for service, e.g., login, are denied or delayed, but transaction sequences already started progress to completion. The forecasting startup approach in Section 4.1 as well as the forecast reset procedure in Section 5.2 solve some of problems users have encountered with Exponential Smoothing in the past. The integer formulation, Section 5.1, permits the algorithm's implementation within an operating system level device driver.

The mathematical formulations and associated illustrations contained in Section 3 thru Section 5 define the algorithm as well as show how and why it works. A thorough understanding of these equations is unnecessary because a computer program that exercises them is provided on [GitHub](#). This C language program, which implements all aspects of the methodology, can be downloaded, compiled, and run.

The primary focus of [US7301903B2](#) is telecommunications products, not computing systems, but if there are patent infringement concerns, the initial filing took place on November 4, 2003 and the grant normally ends 20 years after that filing. There is an extension of 848 days for this patent which makes March 1, 2026 the expiration date.

Although the focus of this paper is congestion management, the algorithm is a generic time series forecaster adaptable to a broad range of prediction situations. Implementation within a computing system monitoring or planning tool to statistically smooth resource consumption or response time service level measurements is one possibility.

8. REFERENCES

- [1] Brady, J.F., 2007. Method and Apparatus for Achieving an Optimal Response Time in A Telecommunications System.

- [2] Brown, R. G., 1963. Smoothing, Forecasting and Prediction of Discrete Time Series. Prentice-Hall, Inc., Englewood Cliffs, N.J.
- [3] Giffin, W. C., 1971. Introduction to Operations Engineering. Irwin, Inc., Homewood Illinois.

APPENDIX A

This appendix shows how Eq. 3 in Section 4.1 is derived by iterating the Eq. 1 Single Exponential Smoothing formula through multiple points in time, t . **Red** is used below to highlight the $t-1$ and $t-2$ time series expressions.

Eq. 1:

$$F_t = S_t(x) = \alpha x_t + (1 - \alpha)S_{t-1}(x)$$

Iterating thru $t-2$:

$$S_t(x) = \alpha x_t + (1 - \alpha)S_{t-1}(x)$$

$$S_{t-1}(x) = \alpha x_{t-1} + (1 - \alpha)S_{t-2}(x)$$

$$S_{t-2}(x) = \alpha x_{t-2} + (1 - \alpha)S_{t-3}(x)$$

⋮

Substituting and expanding:

$$S_t(x) = \alpha x_t + (1 - \alpha)[\alpha x_{t-1} + (1 - \alpha)S_{t-2}(x)]$$

$$S_t(x) = \alpha x_t + \alpha(1 - \alpha)x_{t-1} + (1 - \alpha)^2 S_{t-2}(x)$$

$$S_t(x) = \alpha x_t + \alpha(1 - \alpha)x_{t-1} + (1 - \alpha)^2 [\alpha x_{t-2} + (1 - \alpha)S_{t-3}(x)]$$

$$S_t(x) = \alpha x_t + \alpha(1 - \alpha)x_{t-1} + \alpha(1 - \alpha)^2 x_{t-2} + (1 - \alpha)^3 S_{t-3}(x)$$

⋮

Eq. 3:

$$S_t(x) = \alpha x_t + \alpha(1 - \alpha)x_{t-1} + \dots + \alpha(1 - \alpha)^{i-1}x_{t-i+1} + \dots + (1 - \alpha)^i S_0(x)$$

APPENDIX B

This appendix maps the non-integer Exponential Smoothing equations, Eq. 1 and Eq. 4 thru Eq. 10, to their integer counterparts, Eq. 12 thru Eq. 19. Expressions that are the same in both cases are excluded. The *[integer floor]* bars are used sparingly in the integer expressions below but each component in those expressions is assumed to be implemented as an integer in a C language program using that language's standard operator

precedence where multiplication, addition, and subtraction are performed before division.

For: $0 < \alpha \leq .5$

Let:

$$n_\alpha = \left\lfloor \frac{1}{\alpha} \right\rfloor$$

Then:

$$\frac{\alpha}{(1 - \alpha)} \leq \frac{\frac{1}{n_\alpha}}{(1 - \frac{1}{n_\alpha})} = \frac{\frac{1}{n_\alpha}}{(\frac{n_\alpha - 1}{n_\alpha})} = \frac{1}{(n_\alpha - 1)}$$

F_t : Eq. 4 to Eq. 12

$$F_t = S_t(x) = \frac{1}{n}x_t + \left(1 - \frac{1}{n}\right)S_{t-1}(x)$$

$$F_t = S_t(x) = \frac{1}{n}x_t + \frac{(n-1)}{n}S_{t-1}(x) = \frac{x_t + (n-1)S_{t-1}(x)}{n}$$

b_t : Eq. 9 to Eq. 16

$$b_t = \frac{\alpha}{(1 - \alpha)} [S_t(x) - S_t^{[2]}(x)]$$

$$b_t = \frac{1}{(n_\alpha - 1)} [S_t(x) - S_t^{[2]}(x)] = \frac{[S_t(x) - S_t^{[2]}(x)]}{(n_\alpha - 1)}$$

$S_t(x)$: Eq. 1 to Eq. 17

$$F_t = S_t(x) = \alpha x_t + (1 - \alpha)S_{t-1}(x)$$

$$S_t(x) = \frac{1}{n_\alpha}x_t + \left(1 - \frac{1}{n_\alpha}\right)S_{t-1}(x) = \frac{1}{n_\alpha}x_t + \left(\frac{n_\alpha - 1}{n_\alpha}\right)S_{t-1}(x) = \frac{x_t + (n_\alpha - 1)S_{t-1}(x)}{n_\alpha}$$

$S_t^{[2]}(x)$: Eq. 5 to Eq. 18

$$S_t^{[2]}(x) = \alpha S_t(x) + (1 - \alpha)S_{t-1}S_t^{[2]}(x)$$

$$\begin{aligned} S_t^{[2]}(x) &= \frac{1}{n_\alpha}S_t(x) + \left(1 - \frac{1}{n_\alpha}\right)S_{t-1}^{[2]}(x) \\ &= \frac{1}{n_\alpha}S_t(x) + \left(\frac{n_\alpha - 1}{n_\alpha}\right)S_{t-1}^{[2]}(x) \\ &= \frac{S_t(x) + (n_\alpha - 1)S_{t-1}^{[2]}(x)}{n_\alpha} \end{aligned}$$