The Acceleration of Infinite Time Averaging Methods for Chaotic Systems

Preprint · June 2020

DOI: 10.13140/RG.2.2.36675.63523/2

Imperial College London

Department of Aeronautics

The Acceleration of Infinite Time Averaging Methods for Chaotic Systems

Author
Jamell Ivan Samuels

Supervisor
Dr. Segei Chernyshenko

Second Marker
Dr. Oliver Buxton

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Engineering in Aeronautical Engineering

22 June 2020

Contents

List of Figures

List of Tables

Dedication & Acknowledgements

I would like to dedicate this thesis, to God my family and friends to whom without there love and support I would not have been able to have reached this far in my project. I would also like to personally acknowledge my supervisor for spending many an hour dissecting the work I presented for its further refinement.

Abstract

The calculation of infinite time averages is vital in the study of the general behaviour of chaotic systems. In this thesis we tackle this problem through a variety of methods, both well established and new in order to ascertain the best approach. Due to this field still being in its relative infancy, no definitive method has yet been established to verify and benchmark solutions calculated. In this thesis we attempt to develop new auxiliary functions and establish new theorems to benchmark other approximations by.

Chapter 1

Introduction

1.1 Background and Motivation

Amongst the multitude of natural phenomena encountered by science, one distinct trait recurs within the smallest to largest of these phenomena. Chaotic systems are characterised by their dynamic instability and their apparent random nature. This makes gaining a clear and precise measurement of their parameters virtually impossible. It is therefore given that one must take large to infinite time averages, to eradicate the noise of the fluctuations present in these systems.
As calculating Infinite Time Averages of a chaotic system is a computationally expensive task the desire to accelerate the process is clear.

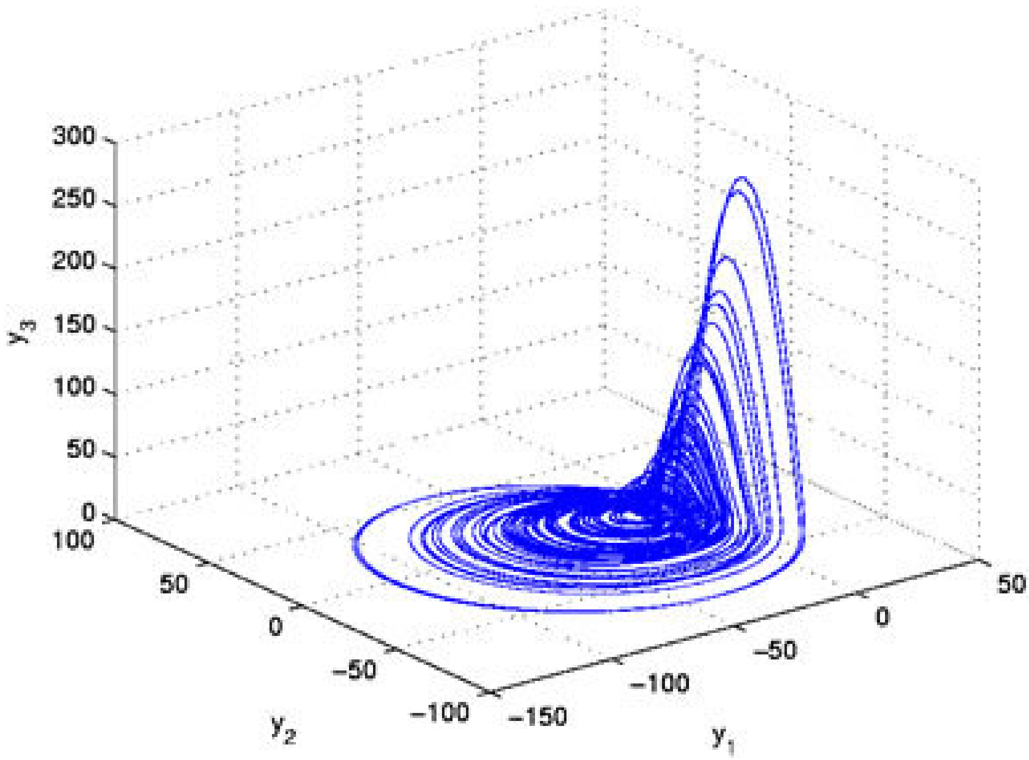

Figure 1.1: Numerical solution of the Rössler attractor [5]

1.2 Concept and Approach

The basis of any infinite time averaging method for chaotic systems, is to solve the systems of ODE's for a suitable time T and than to average by integrating the solution and dividing it by the time T. This process leaves much to be desired as it is computationally very expensive to integrate over a large time T [4]. Although different methods of solving ODE's have been established, our primary focus is on how we shall integrate the chosen system and not on how it shall be solved.

Chapter 2

Systems

The 5 Sprott [3] systems that we shall analyse and plots of their behaviour are detailed below.

2.0.1 Case A

Initial Conditions $\tilde{x} = [x_0, y_0, z_0] = [0.014, 0, -0.014]$.

$$\begin{cases} \frac{dx}{dt} = y \\ \frac{dy}{dt} = -x + yz \\ \frac{dz}{dt} = -1 - y^2 \end{cases} \tag{2.1}$$

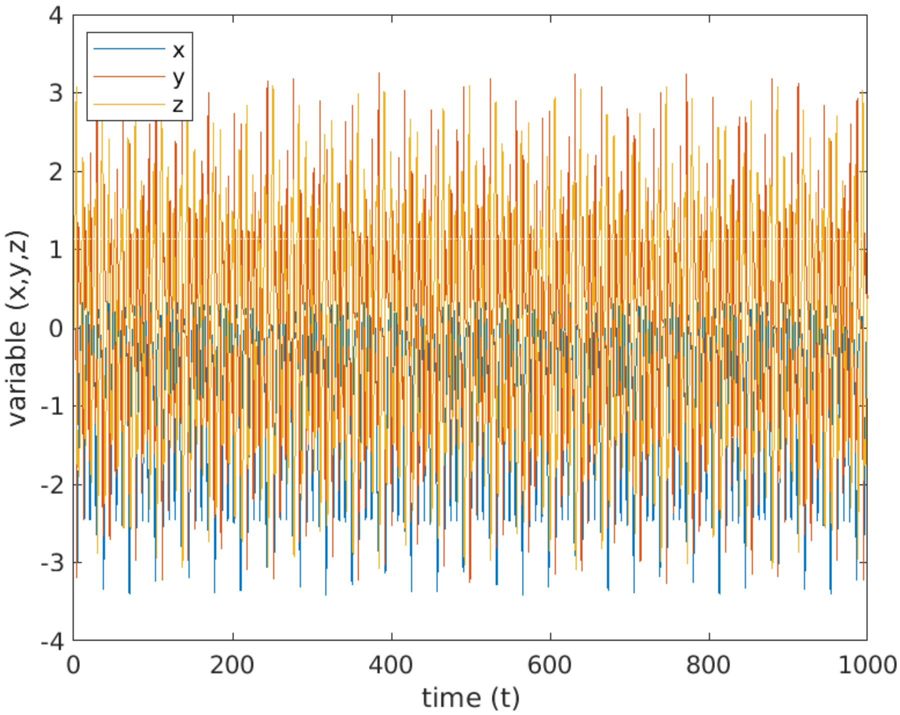

Figure 2.1: Numerical solution of the Case A attractor

The behaviour of x, y, z of Case A is characterised by high frequency oscillations between approximately 3 and -3. The variable y has the greatest oscillation range, while the variable x has the lowest. The overall behaviour of the function doesn't change from $t = 0$ to $T = 1000$ and therefore we can say that the chosen initial conditions were already within the 'periodic sequence' of the system [6].

2.0.2 Case B

Initial Conditions $\tilde{x} = [x_0, y_0, z_0] = [0.210, 0, -0.120]$.

$$\begin{cases} \frac{dx}{dt} = yz \\ \frac{dy}{dt} = x - y \\ \frac{dz}{dt} = 1 - xy \end{cases} \tag{2.2}$$

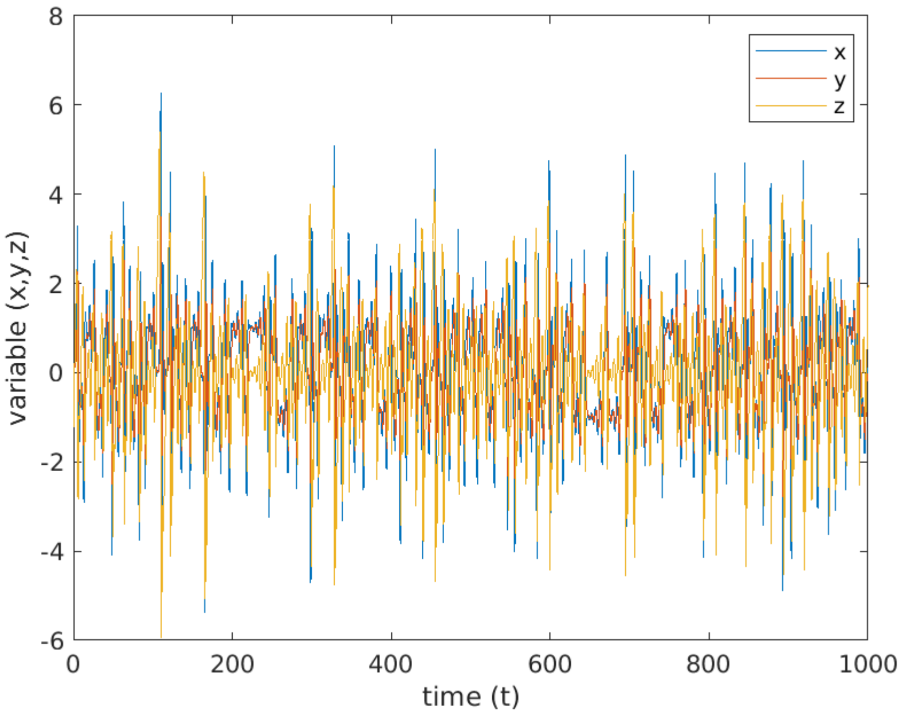

Figure 2.2: Numerical solution of the Case B attractor

The behaviour of Case B is more complex in nature compared to that of case A, as the 'anchor' of the system (the variable that causes the most change) $\frac{dx}{dt}$ is more complex. The system has characteristicly lower frequency fluctuations compared to Case A, however the amplitude of the oscillations and the variation between them is greater, with the maximum amplitude being at $y = 6$ for $T = 100$.

2.0.3 Case C

Initial Conditions $\tilde{x} = [x_0, y_0, z_0] = [0.163, 0, -1.163]$.

$$\begin{cases} \frac{dx}{dt} = yz \\ \frac{dy}{dt} = x - y \\ \frac{dz}{dt} = 1 - xy \end{cases} \tag{2.3}$$

The behaviour of Case C distinguishes itself from that of Case A and Case B, by featuring a more varied response. The visual periodicity has increased and there is a clear pattern where the amplitude of the oscillations will spike before decreasing once again to it's base level. The maximum amplitude is found at approximately $T = 100$, with a y value of approximately 5.8.

2.0.4 Case E

Initial Conditions $\tilde{x} = [x_0, y_0, z_0] = [0.117, 0, -0.617]$.

$$\begin{cases} \frac{dx}{dt} = yz \\ \frac{dy}{dt} = x^2 - y \\ \frac{dz}{dt} = 1 - 4x \end{cases} \tag{2.4}$$

Case E exhibits, an almost uniform frequency distribution not too dissimilar to what you would expect from white noise. It has its peak values at $y \approx 6$, $x \approx 1.2$ $z \approx 1$, with an identifiable pattern of $y \approx 6$ occurring approximately every 100 seconds.

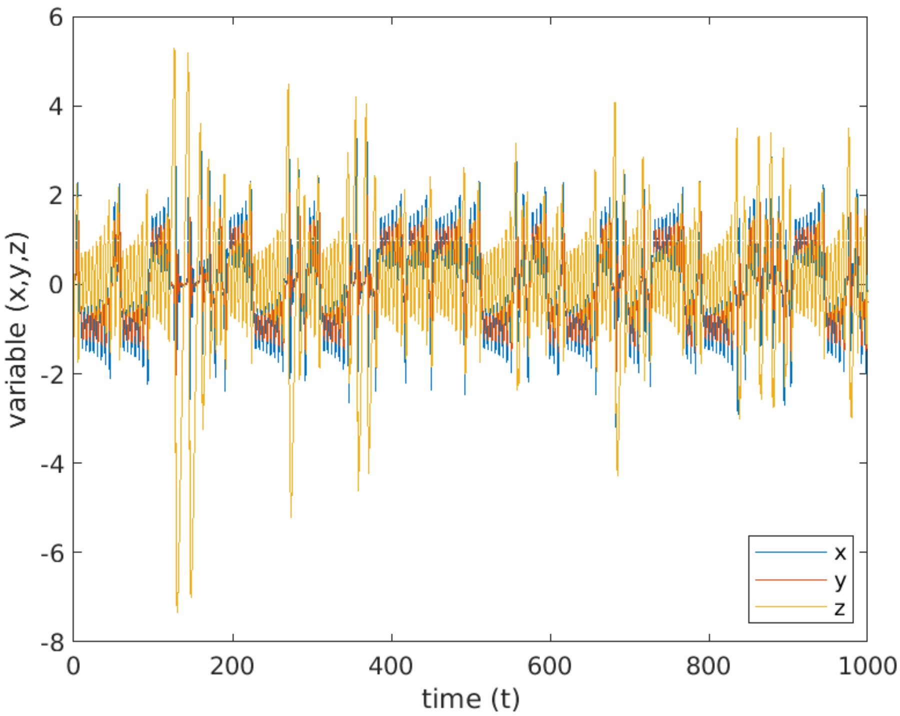

Figure 2.3: Numerical solution of the Case C attractor

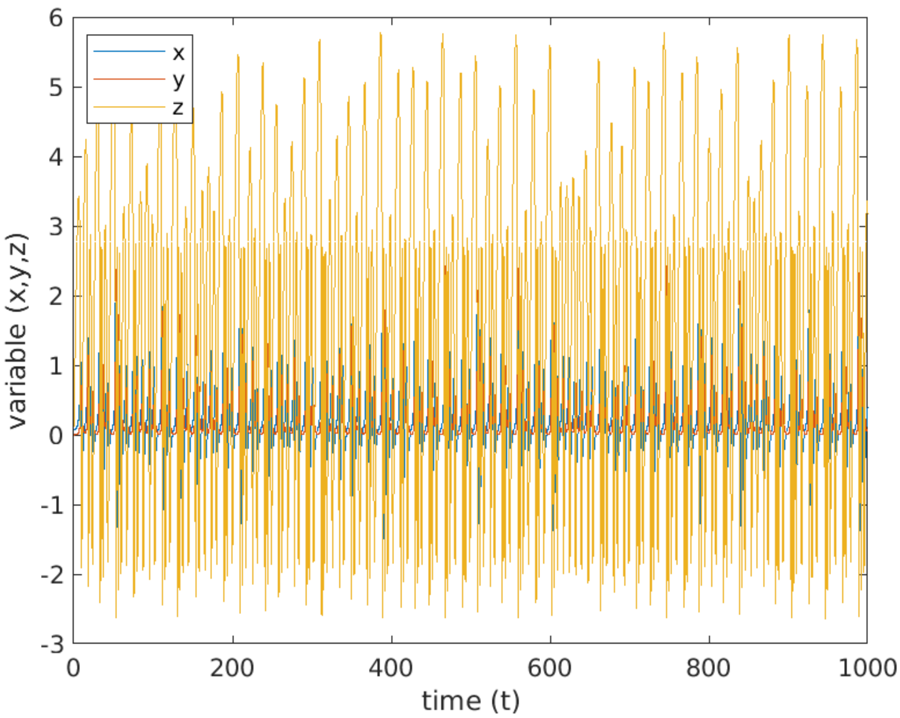

Figure 2.4: Numerical solution of the Case E attractor

2.0.5 Case F

Initial Conditions $\tilde{x} = [x_0, y_0, z_0] = [0.078, 0, 0.117]$.

$$
\begin{cases}
\frac{dx}{dt} = y + z \\
\frac{dy}{dt} = -x + \frac{1}{2}y \\
\frac{dz}{dt} = x^2 - z
\end{cases}
\tag{2.5}
$$

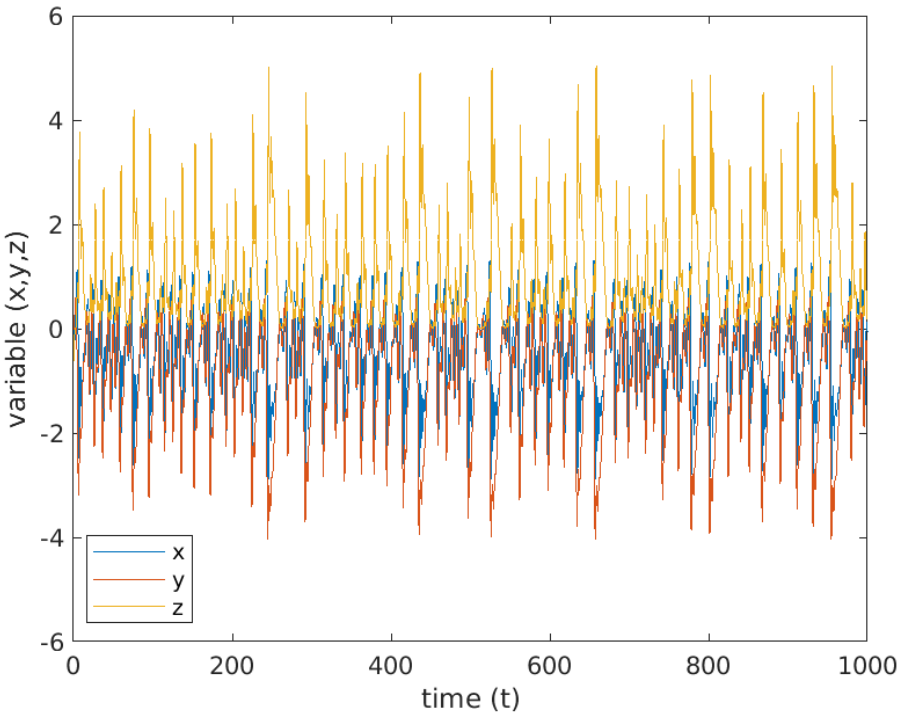

Figure 2.5: Numerical solution of the Case F attractor

Case F is unique compared to the previous cases as it features a split in the direction of amplitude for the y and z values. z is almost exclusively positive with peaks at $z \approx 5$ and y is almost exclusively negative with peaks at $y \approx -4$.

Chapter 3

Theoretical Background

The function that we are looking to time average will be referred to as E. E can be expressed as.

$$E(t) = \frac{1}{2}(x(t)^2 + y(t)^2 + z(t)^2), \tag{3.1}$$

Where $x(t)$,$y(t)$ and $z(t)$ are the states of the system at time t.
The finite time average of E is defined as.

$$\overline{E_T} = \frac{1}{T}\int_0^T E(dt)dt. \tag{3.2}$$

To calculate the infinite time average we apply the limit of $T \to \infty$ to the finite time average.

$$\overline{E_\infty} = \lim_{T\to\infty}\frac{1}{T}\int_0^T E(dt)dt. \tag{3.3}$$

In order to accelerate the averaging of E a previously established technique [1] involving the creation of a new function ϕ is used. $\phi(t)$ is designed to have the same infinite time average as E, however with a reduced variance.

$$\phi(t) = E(t) + \alpha D(t) \equiv \phi = \Sigma_i^N E_i + \alpha_i \frac{dV_i}{dt}. \tag{3.4}$$

Where D is the auxiliary function $\frac{dV}{dt} = \frac{dV}{dx}\frac{dx}{dt}$, α is a scaling co-efficient and V is a special case Sum of Squares polynomial known as a Lyapunov function, constructed to have the following qualities.
$V(x) > 0 \; \forall \; x \; \epsilon \; D$ and $V(0) = 0$ i.e $V(x)$ is positive semi definite and $-\dot{V}(x) = -\frac{\Delta V}{\Delta x}f(x) \geq 0 \forall x \; \epsilon \; \chi$. i.e $\dot{V}(x)$ is a negative semi definite in D. Where $\chi \subset R$[2].

An auxiliary function is any function used to accelerate the convergence of $\overline{E_T}$ to $\overline{E_\infty}$. If $\overline{D_\infty} = 0$ and the result of $\overline{E+D}$ converges faster to the infinite time average than $\overline{E}$ [1], it can be assumed that $(E + D)$ is a faster method for approximating the infinite time average. For this project three different auxiliary functions were chosen.

Type	V_i	D_i
General Auxiliary	$V_1 = x$ $V_2 = y$ $V_3 = z$ $V_4 = xy$ $V_5 = xz$ $V_6 = yz$ $V_7 = xyz$ $V_8 = x^2$ $V_9 = y^2$ $V_{10} = z^2$	$D_1 = \frac{dx}{dt}$ $D_2 = \frac{dy}{dt}$ $D_3 = \frac{dz}{dt}$ $D_4 = x\frac{dx}{dt} + y\frac{dy}{dt}$ $D_5 = x\frac{dz}{dt} + z\frac{dx}{dt}$ $D_6 = y\frac{dz}{dt} + z\frac{dy}{dt}$ $D_7 = xy\frac{dz}{dt} + xz\frac{dy}{dt} + yz\frac{dx}{dt}$ $D_8 = 2x\frac{dx}{dt}$ $D_9 = 2y\frac{dy}{dt}$ $D_{10} = 2z\frac{dz}{dt}$
Specific Auxiliary	$V_1 = x^2$ $V_2 = xy$ $V_3 = xz$ $V_4 = y^2$ $V_5 = yz$ $V_6 = z^2$	$D_1 = 2x\frac{dx}{dt}$ $D_2 = x\frac{dy}{dt} + y\frac{dx}{dt}$ $D_3 = x\frac{dz}{dt}z\frac{dx}{dt}$ $D_4 = 2y\frac{dy}{dt}$ $D_5 = 2y\frac{dz}{dt} + z\frac{dy}{dt}$ $D_6 = 2z\frac{dz}{dt}$
Sine Auxiliary	$V(t) = \frac{E(0)-E(T)}{\pi} sin\frac{E(0)-E(t)}{E(0)-E(T)}\pi$	$D(t) = -\frac{dE(t)}{dt} cos\frac{E(0)-E(t)}{E(0)-E(T)}\pi$

Table 3.1: Auxiliary Functions

As previously stated, the objective of adding any auxiliary function D_i is to reduce the variance of ϕ in comparison to E. The variance of E and ϕ are defined such that.

$$\sigma_E{}^2 = \overline{(E - \overline{E})}. \tag{3.5}$$

$$\sigma_\phi{}^2 = \overline{(E + \alpha_i D_i + \overline{E + \alpha_i D_i})^2}. \tag{3.6}$$

Which can be further expanded to.

$$\sigma_\phi{}^2 = \overline{(E - \overline{E} - \overline{(\alpha D - \overline{\alpha D})})^2} \tag{3.7}$$

$$\sigma_\phi{}^2 = \overline{(E - \overline{E})}^2 + 2\overline{(E - \overline{E})(\alpha D - \overline{\alpha D})} + \overline{(\alpha D - \overline{\alpha D})}^2$$

.

$$\sigma_\phi{}^2 = \sigma_E{}^2 + 2\alpha\overline{D(\phi - \overline{\phi})} + \alpha^2\overline{D^2}$$

.

The final equation is a quadratic of α and therefore will have a minimum α value corresponding to the minimum value of σ_ϕ. This is obtained by setting $\frac{d\sigma_\phi^2}{d\alpha} = 0$.

The expression for α_{min} than becomes.

$$\alpha_{min} = -\frac{\overline{D(\phi - \overline{\phi})}}{\overline{D^2}})$$

.

Alongside the standard average $\overline{E}$, a running time average $\overline{E_{run}}$ was also calculated. $\overline{E_{run}}$ can be defined as.

$$\overline{E_{run}} = \frac{\Sigma_{t=0}^{T} E}{t}(3.8)$$

where t is the current time accumulated.

The Minimum Time Average, is the minimum time an integral has to be calculated for in order to achieve what we expect to be the infinite time average. It is calculated by assuming that chaotic systems like all dynamical systems experience states of complete stops where $|\omega| = 0$, between which the general behaviour of the system is analogous to how the system would behave if ran for an infinite time. A physical derivation and an example of the minimum time equation for case A is displayed below.

$$\omega = [\omega_x, \omega_y, \omega_z] \tag{3.9}$$

$$\omega = \frac{1}{r}\frac{dx}{dt} = \frac{d\phi}{dt}$$

$$\frac{d^2\phi}{dt^2} = \frac{1}{r}\frac{d^2x}{dt^2} = 0$$

Ergo 0 angular and 0 linear net momentum.

$$\frac{d^2x}{dt^2} + \frac{d^2y}{dt^2} + \frac{d^2z}{dt^2} = \frac{dy}{dt} - \frac{dx}{dt} + y\frac{dz}{dt} + z\frac{dy}{dt} - 2y\frac{dy}{dt} \tag{3.10}$$

For the purpose of achieving the most suitable time points to integrate between it is always best to ensure that the maximum and minimum values of $E(t)$ are within the time bounds. This is set by the limit $t_{\frac{d^2\tilde{x}}{dt^2}} <$ $(t)_{E_{minimum}}$ and $\frac{d^2\tilde{x}}{dt^2}{}_{t_2} > E(t)_{maximum}$. where t_1 and t_2 are both stationary points and t of $E(t)_{maximum}$ and t $E(t)_{minimum}$ are points of inflection.

Although Variance and by extension Standard Deviation have been covered , we shall now introduce a similar concept known as Sample Deviation s_d. Sample Deviation is the displacement of a point from the mean μ or $\bar{E}$. This is calculated as $s_d = (x_i - \overline{x})$. By calculating the covariance between the sample deviation and the average, a relationship is established whereby it can be stated that the average at $T \to \infty$ for any system is located where $s_d = 0$.

The derivation for this follows.

$$T \to \infty \;\; Cov(s_d, \overline{E}) \to 0 \tag{3.11}$$

$$\frac{1}{N}\Sigma(\overline{E_T} - \overline{E_\infty})(s_d - \bar{s_d}) \to 0 \tag{3.12}$$

$$T \Rightarrow \infty \;\; \overline{s_d} \rightarrow 0$$

$$\frac{1}{N}(\Sigma(\overline{E_T} - \overline{E_\infty})(s_d) \rightarrow 0$$

$$\Sigma\frac{d\overline{E_T}}{dt}s_d + \Sigma\overline{E_T}\frac{ds_d}{dt} - \Sigma\frac{d\overline{E_\infty}}{dt} - \Sigma\frac{ds_d}{dt}\overline{E_{infty}} \rightarrow 0$$

$$\frac{ds_d}{dt} = 0, \;\; \frac{d\overline{E_\infty}}{dt} = 0$$

$$\Sigma\frac{dE}{dt}s_d \rightarrow 0$$

$$\Sigma_1^N s_d \rightarrow 0.$$

This can than be used to find the value N which satisfies the above condition, which can than be used to find the corresponding running average E_{sd}.

An alternate derivation using standard deviation is included in the appendix.

Chapter 4

Case B Study

Although the investigation was conducted on 5 different systems, we shall only take an in depth look into one case. The results for the rest of the cases can be found in the appendix. Case B was chosen for its typical behaviour.

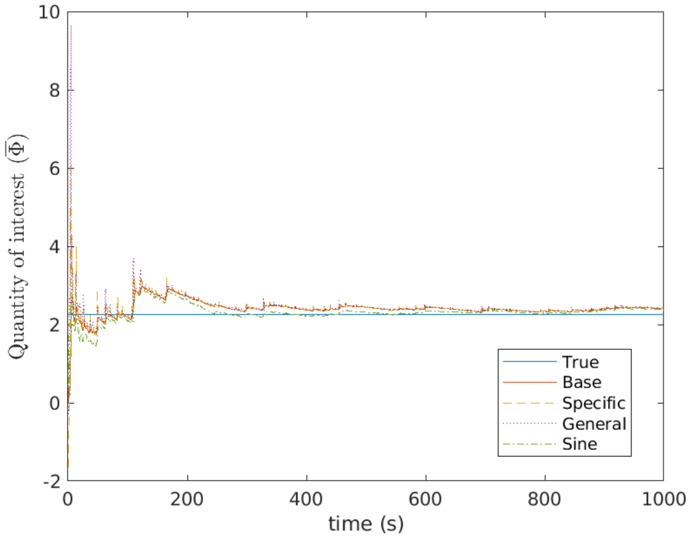

Figure 4.1: Case - B Base Averages

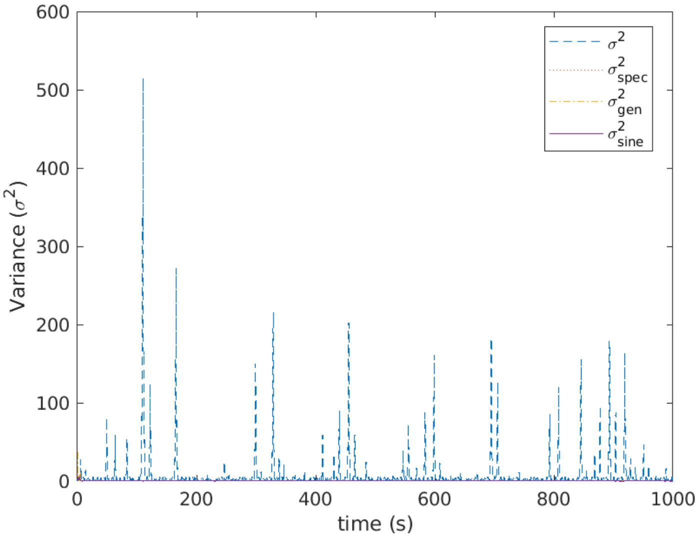

Figure 4.2: Case - B Minimum Time Base Averages

The first study shows the evolution of the base $\overline{E}$ alongside the calculated $\overline{\phi}$'s which were $\overline{E + D_{gen}}$, $\overline{E + D_{spec}}$ and $\overline{E + D_{sine}}$. It can clearly be seen that the averages follow each other closely, showing little to no difference between them apart from the initial spike in $\overline{E + D_{gen}}$, where the value of the average was grossly overestimated compared to the other auxiliary functions.

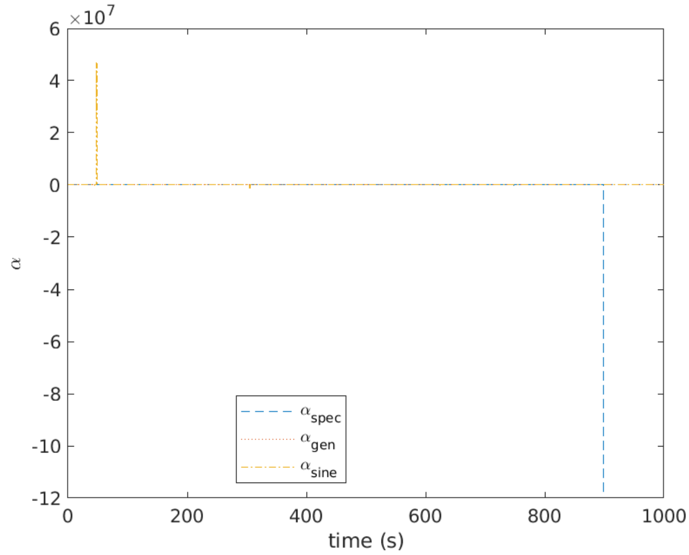

Figure 4.3: Case - B α_{min}

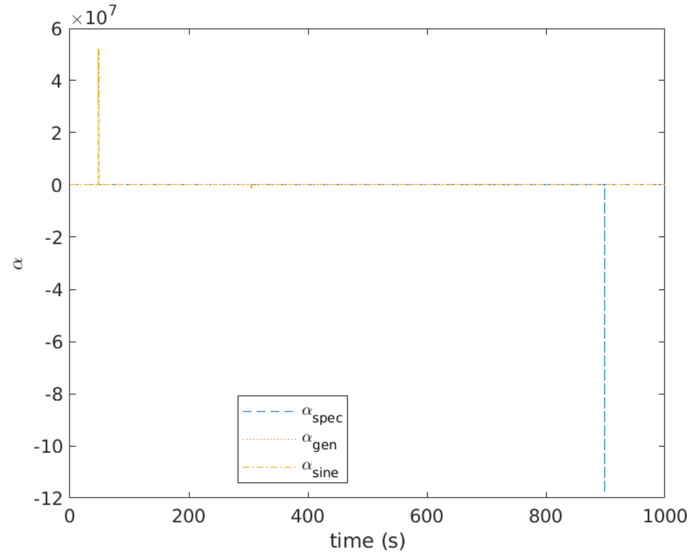

Figure 4.4: Case - B α_{run}

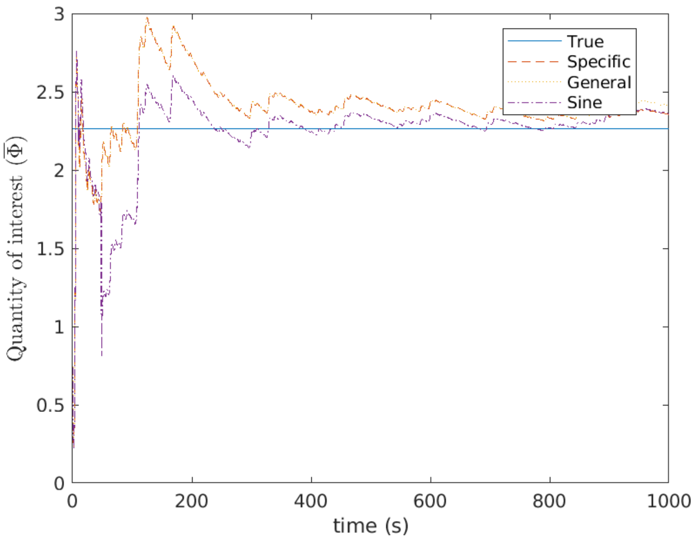

Figure 4.5: Case - B α_{min} Averages

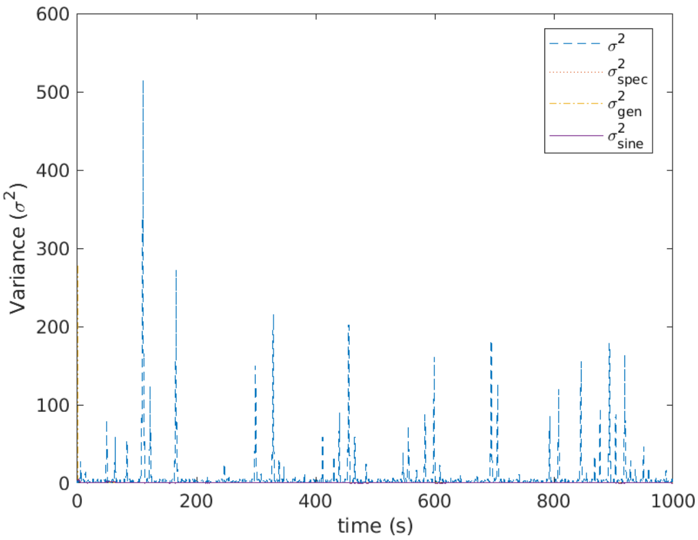

Figure 4.6: Case - B α_{min} Variances

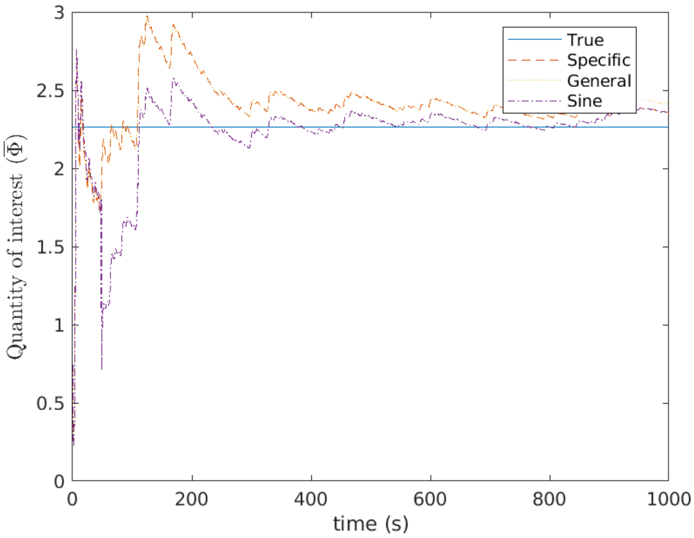

Figure 4.7: Case - B α_{run} Averages

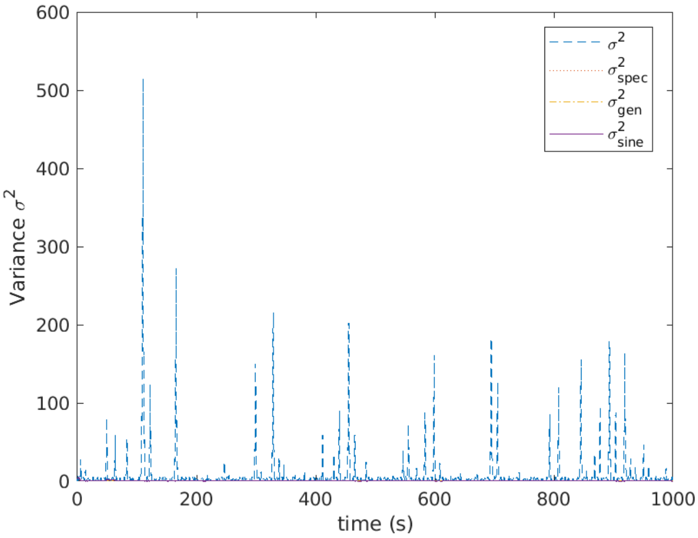

Figure 4.8: Case - B α_{run} Variances

Graphically, the α_{min} and α_{run} averages are very similar. All the auxiliary cases spike initially before dropping. The sine case exhibits the largest drop before recovering to become the closest to $\overline{E_\infty}$. The general case was the worst of the three cases overestimating the value of $\overline{E_\infty}$ and the specific case was the most consistent.

The behaviour shown by the averages shows an overall reduction in the variance when an auxiliary function is applied. It can be seen that the application of an α is important in reducing the variance. The general case shows that the addition of an auxiliary function alone will not guarantee a reduction in variance, however it can be said that certain auxiliary functions such as the sinusoidal function will achieve a reduction in variance without the added need of an α.

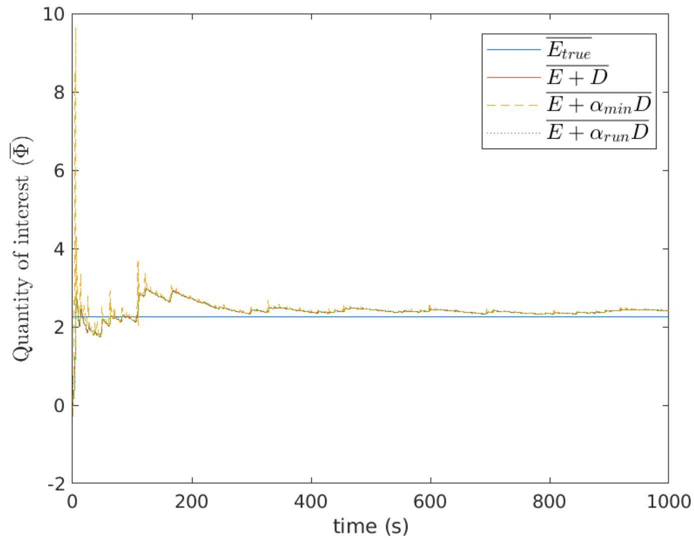

Figure 4.9: Case - B General Auxiliary Averages

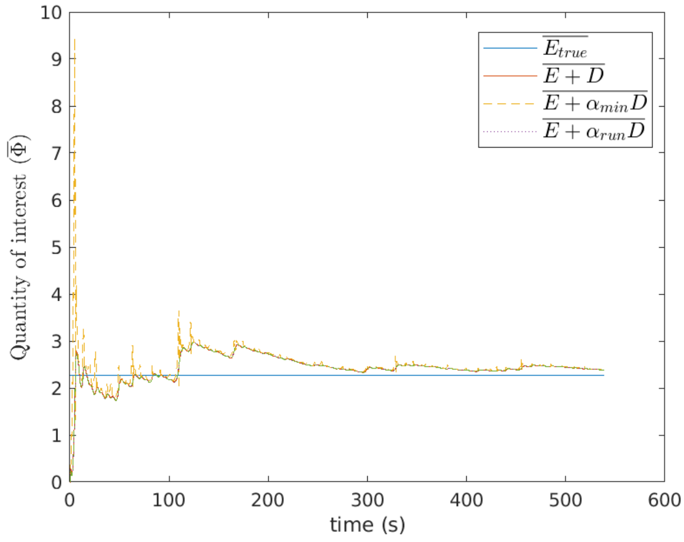

Figure 4.10: Case - B Minimum Time General Auxiliary Averages

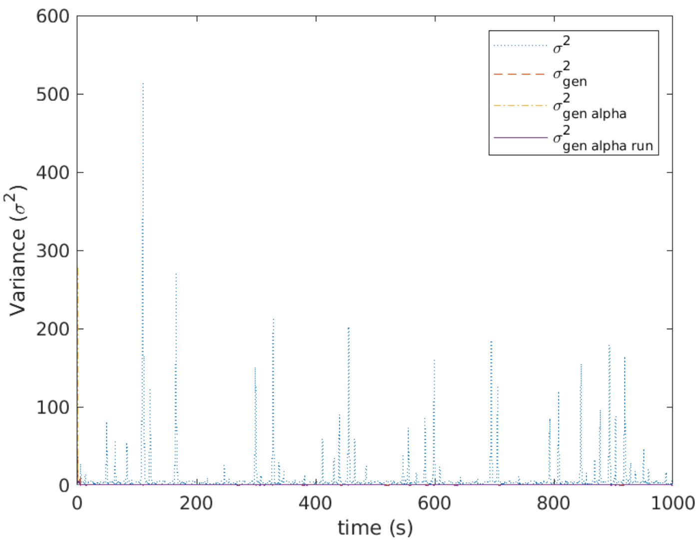

Figure 4.11: Case - B General Auxiliary Variances

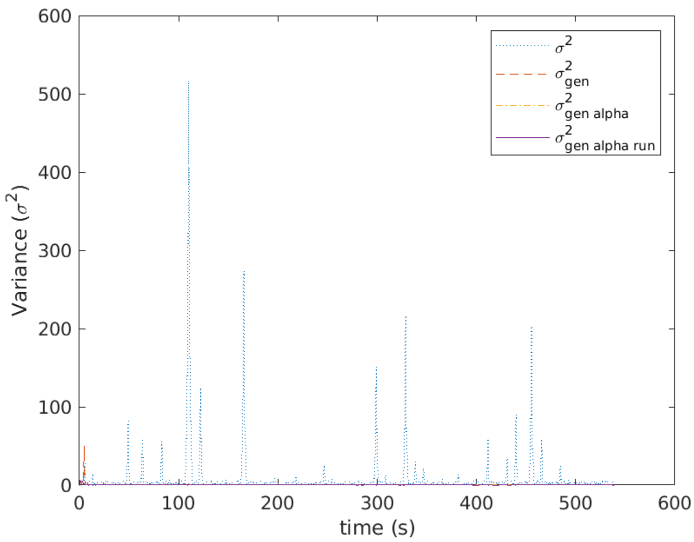

Figure 4.12: Case - B Sine Auxiliary Averages

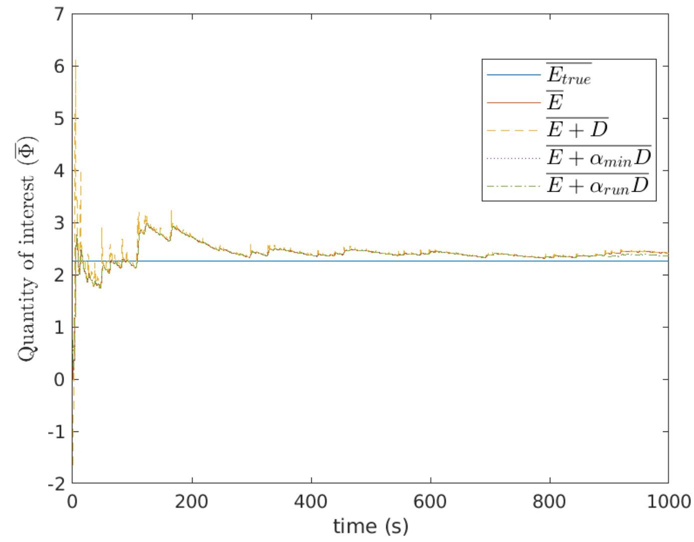

Figure 4.13: Case - B Specific Auxiliary Averages

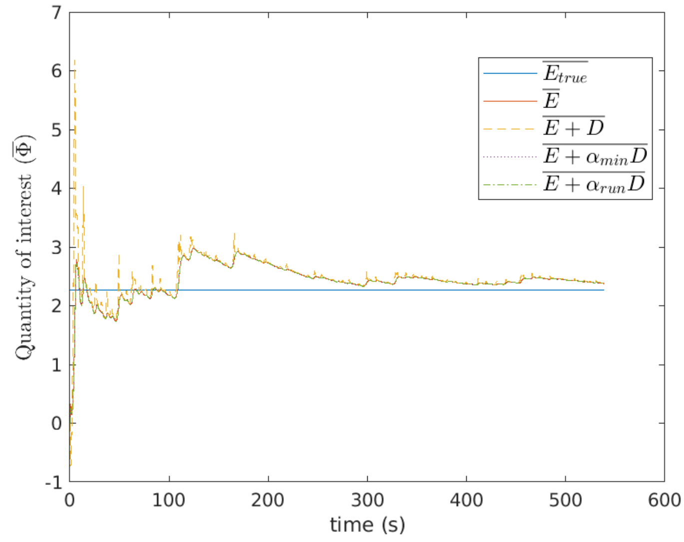

Figure 4.14: Case - B Minimum Time Specific Auxiliary Averages

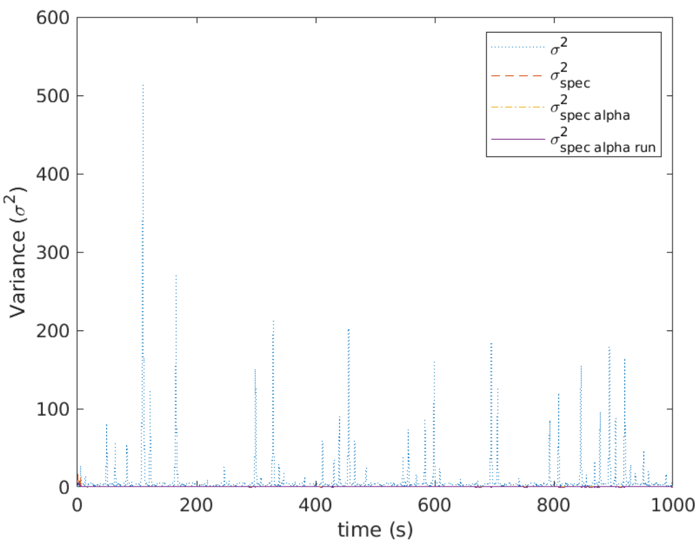

Figure 4.15: Case - B Specific Auxiliary Variances

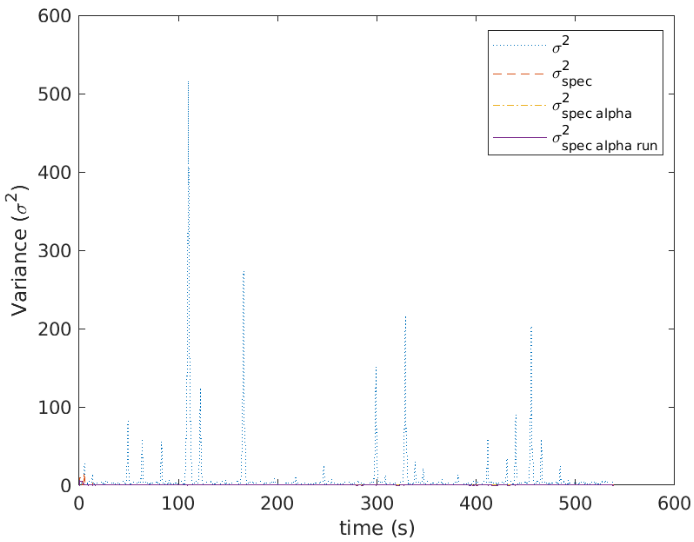

Figure 4.16: Case - B Minimum Time Specific Auxiliary Variances

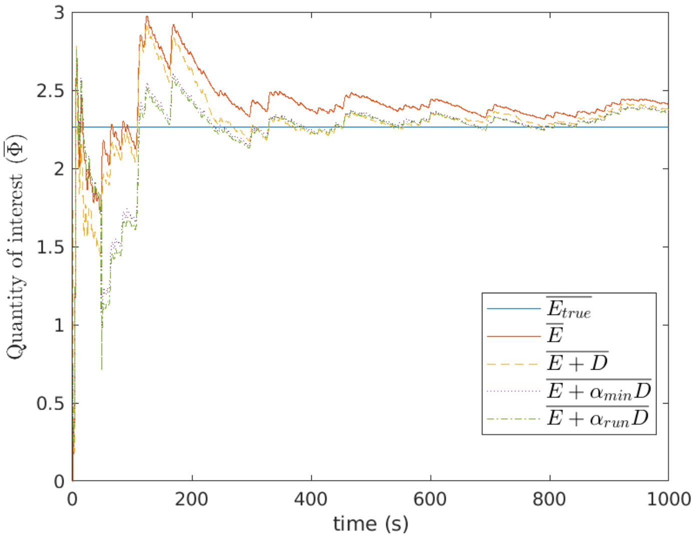

Figure 4.17: Case - B Sine Auxiliary Averages

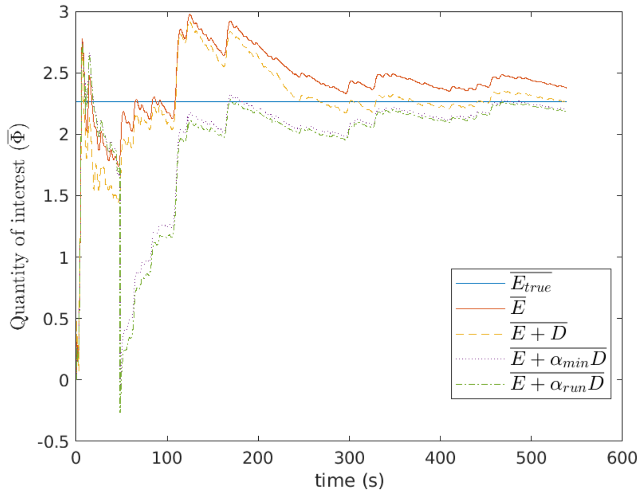

Figure 4.18: Case - B Minimum Time Sine Auxiliary Averages

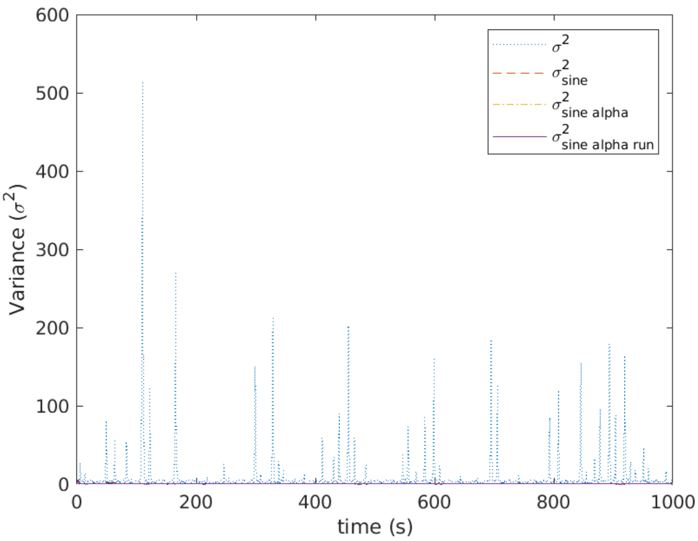

Figure 4.19: Case - B Sine Auxiliary Variances

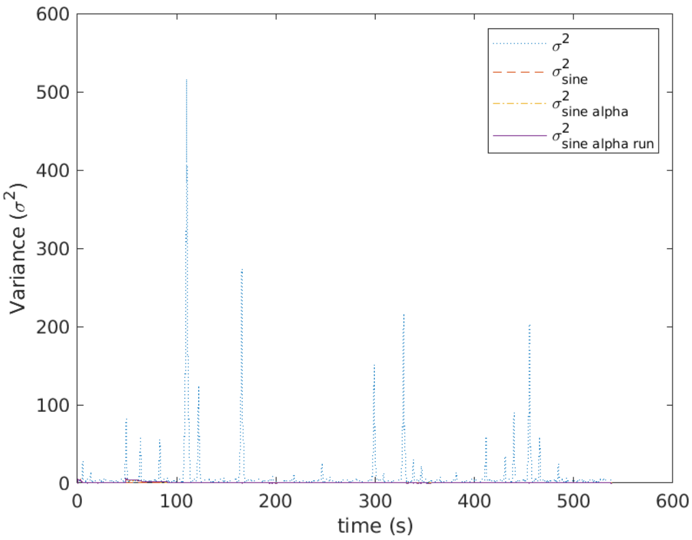

Figure 4.20: Case - B Minimum Time Sine Auxiliary Averages

The minimum time cases demonstrate slightly more erratic behaviour before settling. It can be seen that the use of an α is more important when calculating minimum time due to the increase in fluctuations. The sine case displayed the most erratic behaviour when calculated under minimum time, however it also became one of the more accurate approximations once given sufficient running time. All minimum times were calculated as being over half of the full running time. It can than be suggested, that a larger running time be chosen, so that a wider array of points can be used.

Chapter 5

Results and Discussion

The results for the 5 Sprott systems are tabulated below.

Case	$\overline{E}$	$\overline{E+D_{gen}}$	$\overline{E+D_{spec}}$	$\overline{E+D_{sine}}$	$\overline{E_{sd}}$	$\overline{E_{T=10000}}$	$\overline{E_{T=100000}}$
A	2.2554	3.1306	2.2554	2.2735	2.2474	2.2524	2.2528
B	2.4075	2.4125	2.4076	2.3742	2.3446	2.2634	2.2638
C	1.9175	1.9134	1.9137	1.9761	2.0341	1.8645	1.9066
F	2.3840	2.3841	2.3840	2.3672	2.2587	2.4272	2.3586
E	4.3462	4.3625	4.3642	4.6829	4.2670	4.1671	4.1803

Table 5.1: Case Averages

From the results tabulated it can be seen that E_{sd} is the best approximation for an infinite time average when compared to the base axillary functions of D. Out of the auxiliary functions D_{sine} was the better approximation being closer to the true average $E_{T=100000}$ than D_{spec} howeverD_{sine} has a slightly higher error tolerance as can be seen in Case A.

Case	$\overline{E}$	$\overline{E+\alpha_{min}D_{gen}}$	$\overline{E+\alpha_{min}D_{spec}}$	$\overline{E+alpha_{min}D_{sine}}$	$\overline{E_{sd}}$	$\overline{E_{T=10000}}$	$\overline{E_{T=100000}}$
A	2.2554	2.5494	2.2554	2.2540	2.2474	2.2524	2.2528
B	2.4075	2.4085	2.4076	2.3582	2.3446	2.634	2.2638
C	1.9175	1.9123	1.9137	1.6244	2.0341	1.8645	1.9066
F	2.3840	2.3814	2.3840	2.3622	2.2587	2.4272	2.3586
E	4.3462	4.3461	4.3462	4.3467	4.2670	4.1671	4.1803

Table 5.2: Case α_{min} Averages

After multiplying the auxiliary function by α_{min}, E_{sd} can still be considered to be the most accurate approximation. However, the auxiliary functions do display more accuracy when compared to $\overline{E_T}$ for certain cases. The auxiliary functions show very similar results in this study, as once attenuated by α the general and sine cases have gained a degree of accuracy equivalent to the specific.

Case	$\overline{E}$	$\overline{E+\alpha_{run}D_{gen}}$	$\overline{E+\alpha_{run}D_{spec}}$	$\overline{E+alpha_{run}D_{sine}}$	$\overline{E_{sd}}$	$\overline{E_{T=10000}}$	$\overline{E_{T=100000}}$
A	2.2554	2.5494	2.2553	2.2741	2.2474	2.2524	2.2528
B	2.4075	2.4085	2.3527	2.3534	2.3446	2.2634	2.2638
C	1.9175	1.9123	1.9122	1.7168	2.0341	1.8645	1.9066
F	2.3840	2.3814	2.3794	2.3622	2.2587	2.4272	2.4272
E	4.3462	4.3461	4.3462	4.3467	4.2670	4.1671	4.1803

Table 5.3: Case α_{run} Averages

5.0.1 Minimum Time

In general the minimum time approximations faired worse than the full time study.It was only in case A that the minimum time approximation out did that of the full time. This could be due to a caveat in the code, where if the full conditions could not be met within the time period set, the minimum or maximum values of E were used instead of any stationary points. This most likely led to an overestimation of the average, compared to what it should have been. We can therefore conclude, that the minimum time approximation should be far more powerful if given a larger running time than $T = 1000s$.

Case	$\overline{E_{T=100000}}$	$\overline{E+D_{gen}}$	$\overline{E+D_{spec}}$	$\overline{E+D_{sine}}$	$\overline{E_{minT}}$	$\overline{E+D_{gen_{minT}}}$	$\overline{E+D_{spec_{minT}}}$	$\overline{E+D_{sineminT}}$
A	2.2528	3.1306	2.2554	2.2735	2.5360	3.1252	1.6893	2.2720
B	2.2638	2.4125	2.4076	2.3742	2.4585	2.4613	2.4670	2.2533
C	1.9066	1.9134	1.9137	1.9761	2.3262	2.3521	2.3381	2.4896
F	2.3586	2.3841	2.3840	2.3672	2.3636	2.4074	2.3857	2.4022
E	4.1803	4.3625	4.3642	4.6829	4.3416	4.3549	4.3486	4.6025

Table 5.4: Base vs. Minimum Time Base Comparison

Case	$\overline{E_{T=100000}}$	$\overline{E+\alpha D_{gen}}$	$\overline{E+\alpha D_{spec}}$	$\overline{E+\alpha D_{sine}}$	$\overline{E_{minT}}$	$\overline{E+\alpha D_{gen_{minT}}}$	$\overline{E+\alpha D_{spec_{minT}}}$	$\overline{E+\alpha D_{sineminT}}$
A	2.2528	2.5494	2.2554	2.2540	2.5360	2.2530	2.2543	2.2741
B	2.2638	2.4085	2.4076	2.3582	2.4585	2.4585	2.4579	2.1966
C	1.9066	1.9123	1.9137	1.6244	2.3262	2.3265	2.2786	2.3474
F	2.3586	2.3814	2.3840	2.3622	2.3636	2.3632	2.3573	2.3938
E	4.1803	4.3461	4.3462	4.3467	4.3416	4.3425	4.3421	4.2699

Table 5.5: α_{min} vs. Minimum Time α_{min} Comparison

Case	$\overline{E_{T=100000}}$	$\overline{E+\alpha D_{gen}}$	$\overline{E+\alpha D_{spec}}$	$\overline{E+\alpha D_{sine}}$	$\overline{E_{minT}}$	$\overline{E+\alpha D_{gen_{minT}}}$	$\overline{E+\alpha D_{spec_{minT}}}$	$\overline{E+\alpha D_{sineminT}}$
A	2.2528	2.5494	2.2553	2.2740	2.5360	2.2530	2.2543	2.2514
B	2.2638	2.4085	2.3527	2.3534	2.4585	2.4585	2.4579	2.1819
C	1.9066	1.9123	1.9122	1.7168	2.3262	2.3265	2.2786	2.3334
F	2.3586	2.3814	2.3794	2.3622	2.3636	2.3632	2.3573	2.3940
E	4.1803	4.3461	4.3462	4.3467	4.3416	4.3425	4.3421	4.2699

Table 5.6: α_{run} vs. Minimum Time α_{run} Comparison

5.0.2 Error Calculation

Errors were calculated, however due to the time constraints on the project, only the α_{run} errors shall compared.

Case	$\overline{E}$	$\overline{E+\alpha D_{gen}}$	$\overline{E+\alpha D_{spec}}$	$\overline{E+\alpha D_{sine}}$	$\overline{E_{sd}}$
A	0.0012	0.1317	0.0011	0.1257	-0.0024
B	0.0635	0.0639	0.0393	0.0396	0.0357
C	0.0057	0.0030	0.0029	-0.0995	0.0669
F	0.0108	0.0097	0.0088	0.0015	-0.0424
E	0.0440	0.0397	0.0397	0.0398	0.0207

Table 5.7: α_{run} case errors

Case	$\overline{E}_{minT}$	$\overline{E+\alpha D_{gen}}_{minT}$	$\overline{E+\alpha D_{spec}}_{minT}$	$\overline{E+\alpha D_{sine}}_{minT}$
A	0.1257	$8.877x10^{-5}$	$6.658x10^{-4}$	$-6.2145x10^{-4}$
B	0.0860	0.0860	0.0857	-0.0362
C	0.2201	0.2202	0.1951	0.2239
F	0.0386	0.0388	0.0387	0.0214
E	0.0386	0.0388	0.0387	0.0214

Table 5.8: Minimum Time α_{run} case errors

From analysing the errors, a number of conclusions can be drawn. If considering both the full running time and the minimum running time, the most accurate method overall was the specific auxiliary function, With a total error of 0.4507. For the minimum time however, the most accurate method, was the general auxiliary function with a total error of 0.3839. However, if not considering case A, which was very accurate across the board, the most accurate method was the sine case, followed by the specific case with errors of 0.3029 and 0.3582.

Chapter 6

Conclusion

The study of infinite time averaging is a relatively new field in the study of Chaos. In this paper we conducted an investigation using previously established methods, as well as deriving our own minimum time and running distribution methods. The general trend of the data collected showed a fluctuation of the average around E_{sd}, ("Converging Distribution" - See Appendix). Hinting towards the establishment of a theoretical average. Further study would be needed into the implementation of the minimum time approach, but the results show promise when used in tandem with an auxiliary function. Of the auxiliary functions used the specific function offered the most consistent approximation for the infinite time average, however the sine function performed better under less time. It should be noted however that $\overline{D} \neq 0 \ \forall \ T$ and therefore the average gained is not actually the average of the intended function. It should therefore be concluded that the best approach be a running/converging distribution method.

Bibliography

[1] Chernyshenko S.,Tutty O., Yang., H, "Accelerating time averaging" *Draft*, April 2019

[2] Papachristodoulou A., Prajna S., "A Tutorial on Sum of Squares Techniques for Systems Analysis" *2005 American Control Conference*, 2005

[3] Sprott J.C., "Some Simple Chaotic Flows" *Physical Review E*, 1994

[4] Aden A., "Accelerating Time Averaging" *Imperial College*, May 2019

[5] Sun Z., Yang X., "Parameters Identification and Synchronization of Chaotic Delayed Systems Containing Uncertainties and Time-Varying Delay" *Mathematical Problems in Engineering* May 2010

[6] Sobottka M., Oliveira L.P.I, "Periodicity and Predictability in Chaotic Systems" *The American Mathematical Monthly*

Chapter 7

Appendices

7.1 Standard Deviation Derivation

$$T \to \infty \;\; Cov(\sigma^2, \overline{E}_T) \to 0$$

$$\frac{1}{N-1}\Sigma(\overline{E}_T - \overline{E}_\infty)(\sigma_T^2 - \overline{\sigma^2}_T) \to 0$$

$$\overline{\sigma_T^2} = 0 \;\; \forall T$$

$$\Sigma(\overline{E}_T - \overline{E}_\infty)\sigma_T^2 \to 0$$

$$\Sigma\overline{E}_T\frac{d\sigma_E^2}{dt} + \Sigma\sigma_T^2\frac{\overline{E_T}}{dt} - \Sigma\overline{E_\infty\frac{d\sigma}{dt}} - \Sigma\sigma_T^2\frac{d\overline{E}_\infty}{dt} \to 0$$

$$T \to \infty \;\; \frac{d\overline{E_\infty}}{dt} = 0 \;\; \frac{d\sigma_T^2}{dt} = 0$$

variance in time does not change for a chaotic system.

$$\therefore$$

$$\Sigma\sigma_T^2\frac{d\overline{E_T}}{dt} \to 0$$

Of note.

$$\sigma = \frac{1}{N}\Sigma s_d^2$$

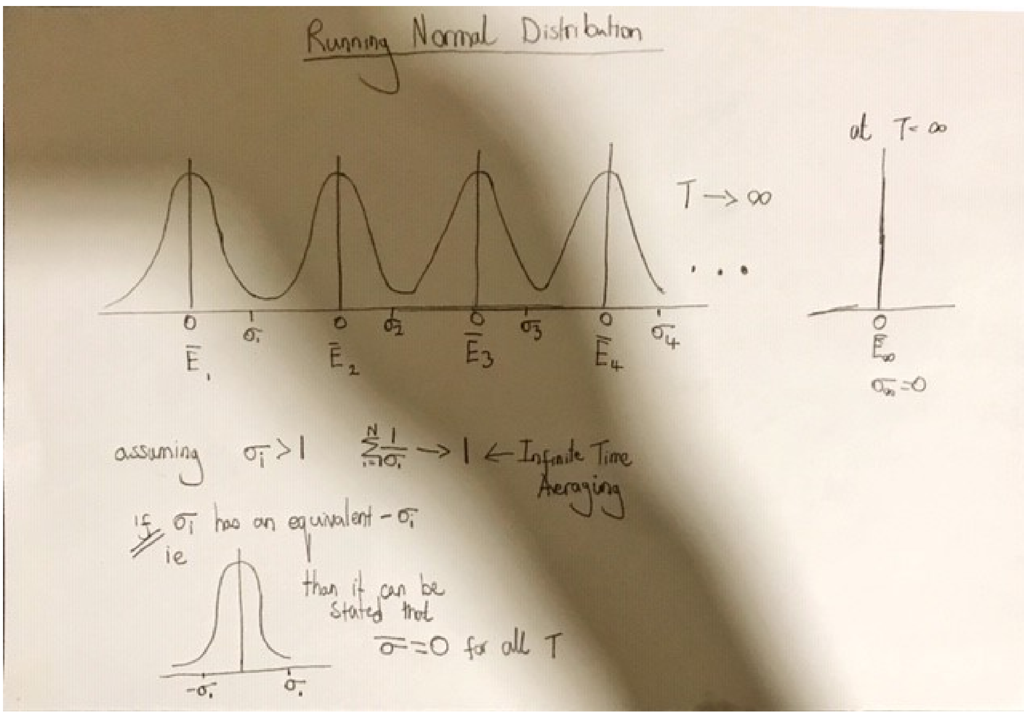

Figure 7.1: Converging Distribution

correction - there's a small mistake regarding the convergence of σ to 1. It converges to ∞. regardless it doesn't matter.

7.2 Plots

7.2.1 Case A

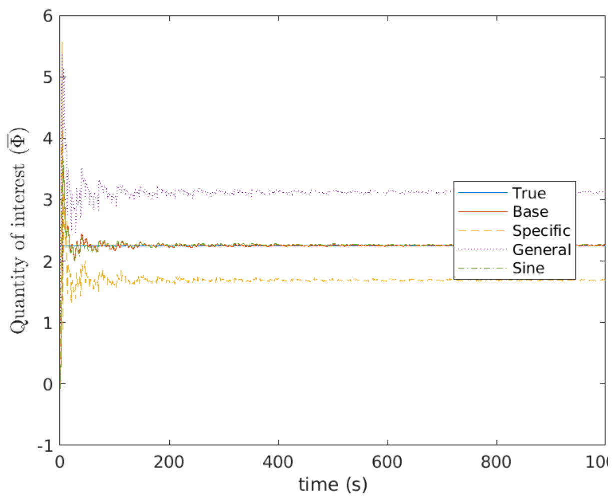

Figure 7.2: Case - A Base Averages

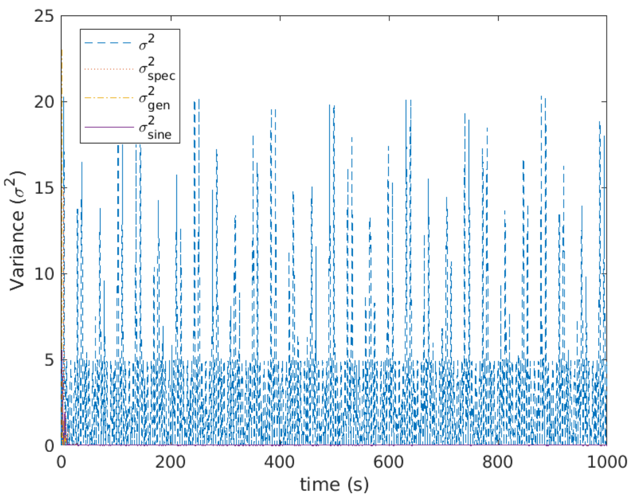

Figure 7.3: Case - A Minimum Time Base Averages

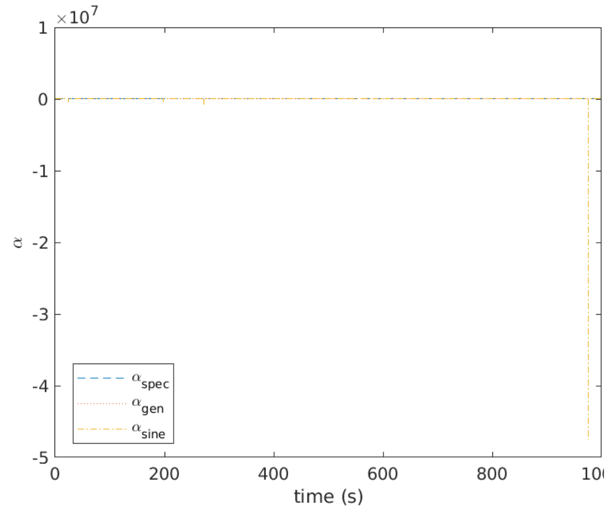

Figure 7.4: Case - A α_{min}

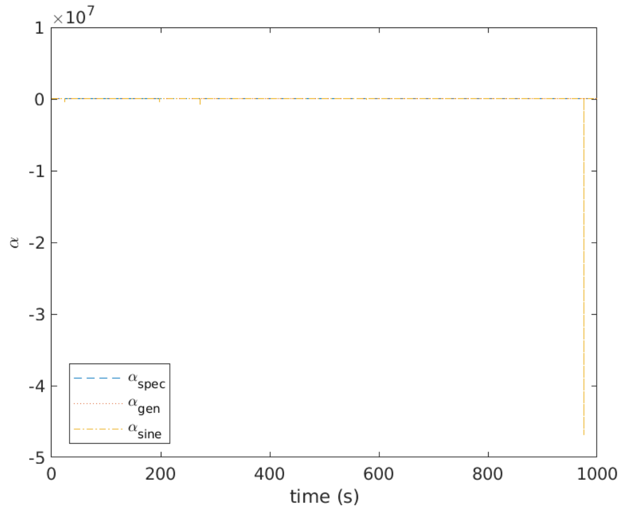

Figure 7.5: Case - A α_{run}

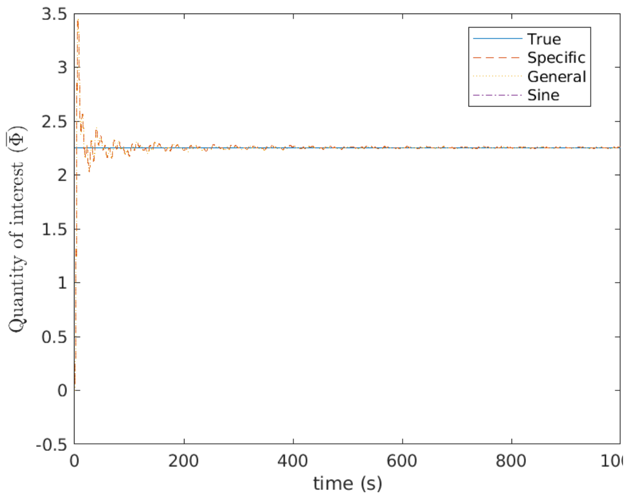

Figure 7.6: Case - A α_{min} Averages

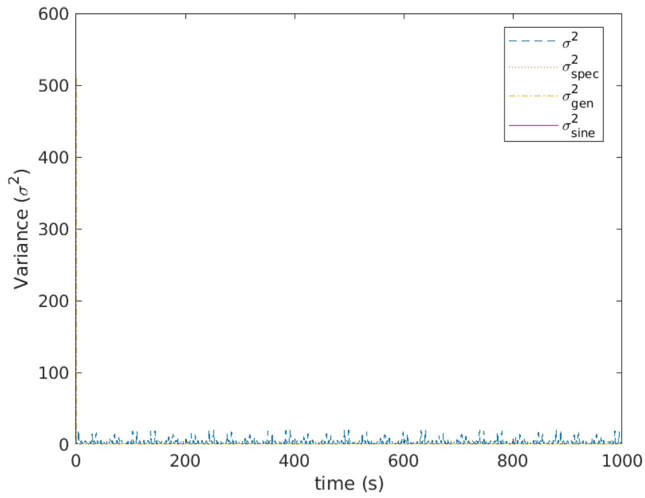

Figure 7.7: Case - A α_{min} Variances

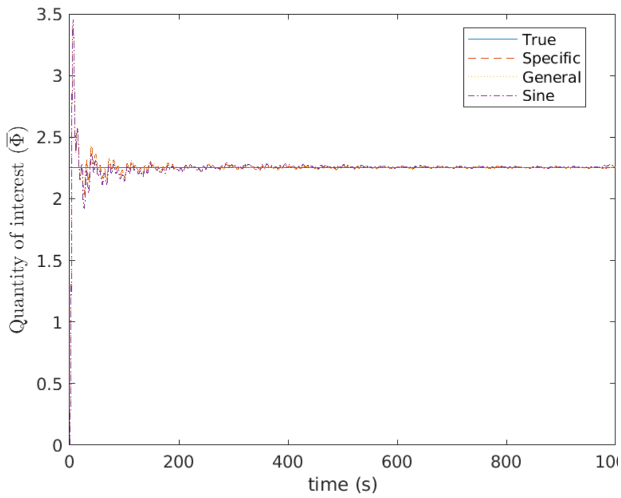

Figure 7.8: Case - A α_{run} Averages

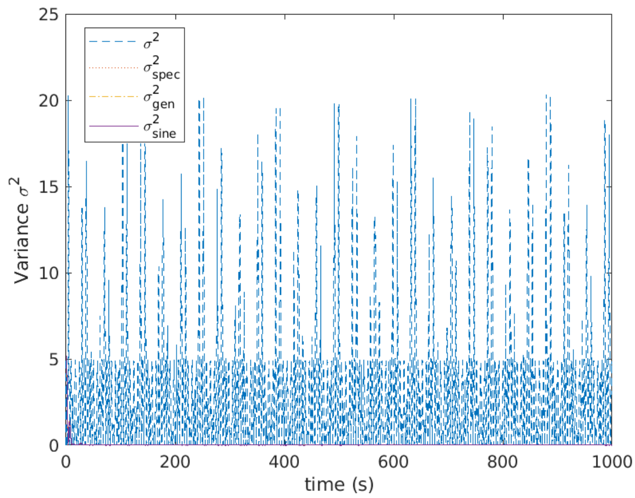

Figure 7.9: Case - A α_{run} Variances

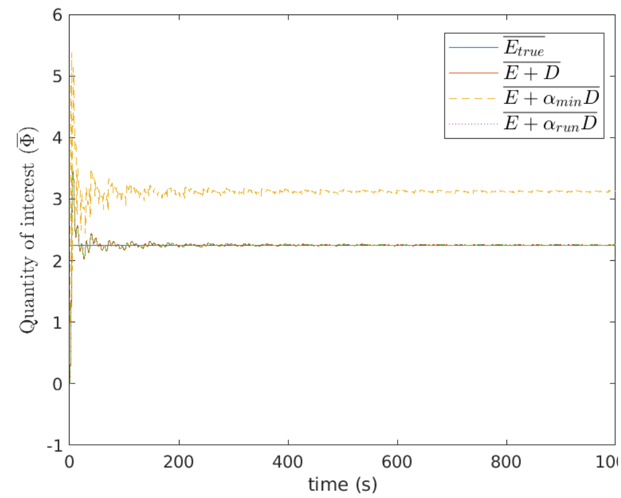

Figure 7.10: Case - A General Auxiliary Averages

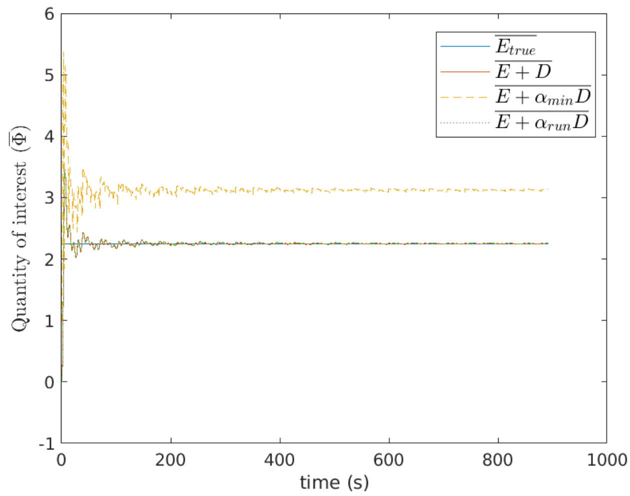

Figure 7.11: Case - A Minimum Time General Auxiliary Averages

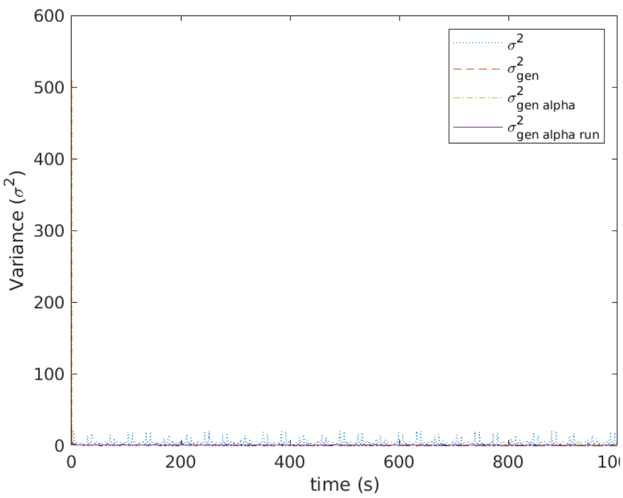

Figure 7.12: Case - A General Auxiliary Variances

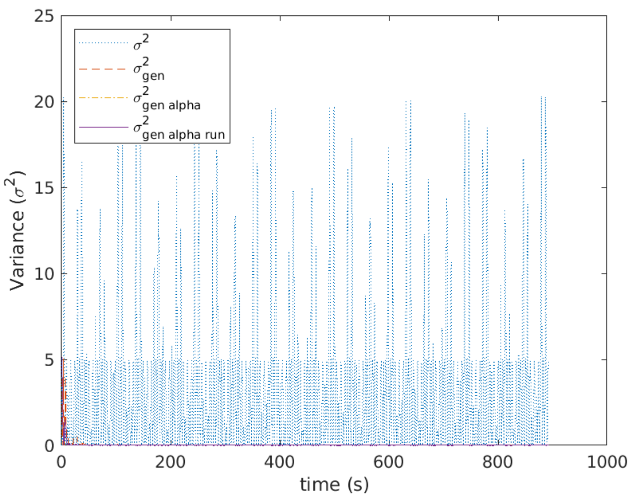

Figure 7.13: Case - A Sine Auxiliary Averages

7.2.2 Case C

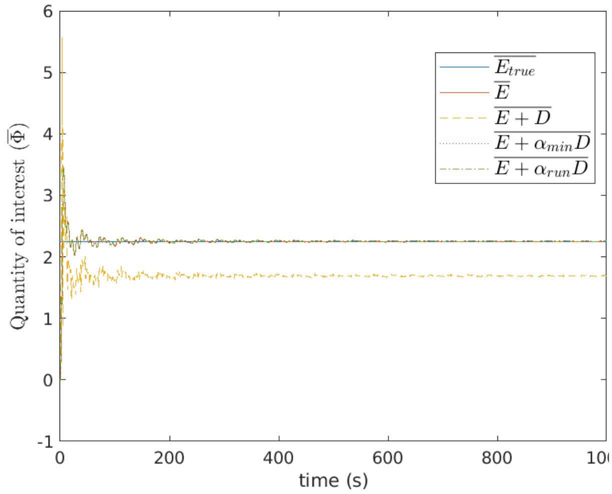

Figure 7.14: Case - A Specific Auxiliary Averages

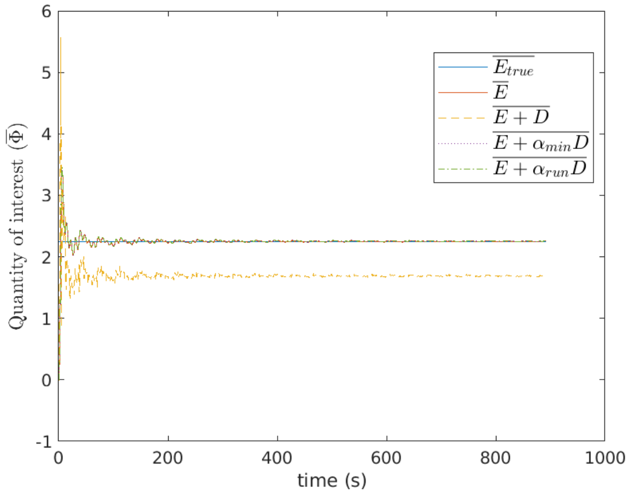

Figure 7.15: Case - A Minimum Time Specific Auxiliary Averages

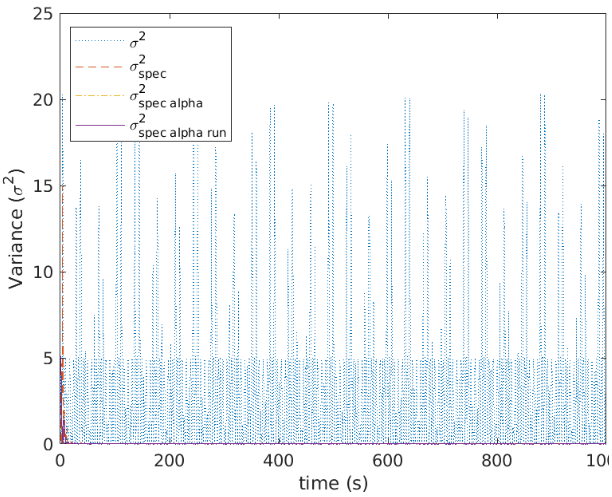

Figure 7.16: Case - A Specific Auxiliary Variances

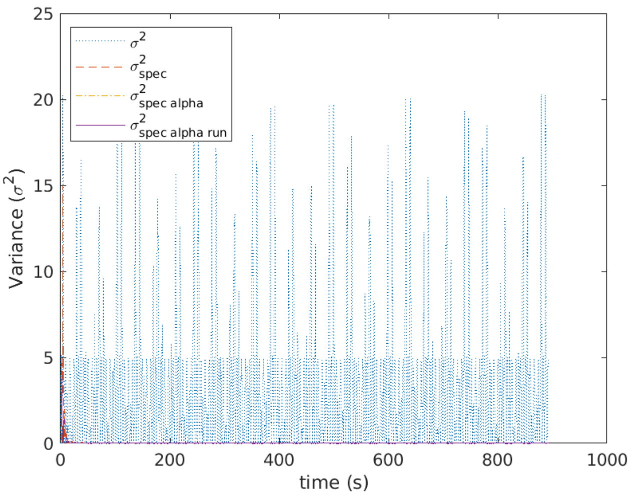

Figure 7.17: Case - A Minimum Time Specific Auxiliary Variances

7.2.3 Case E

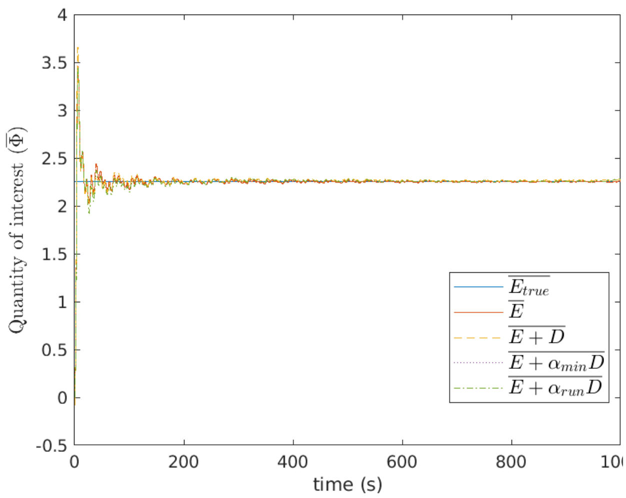

Figure 7.18: Case - A Sine Auxiliary Averages

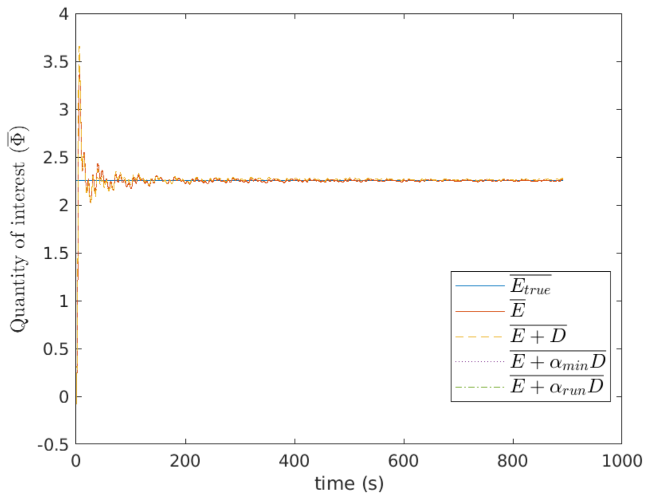

Figure 7.19: Case - A Minimum Time Sine Auxiliary Averages

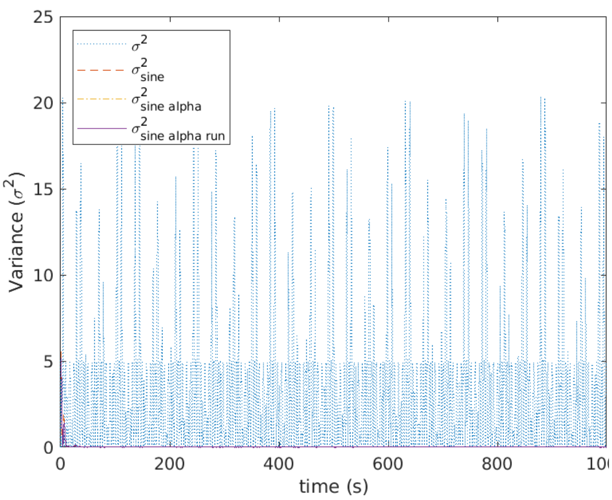

Figure 7.20: Case - A Sine Auxiliary Variances

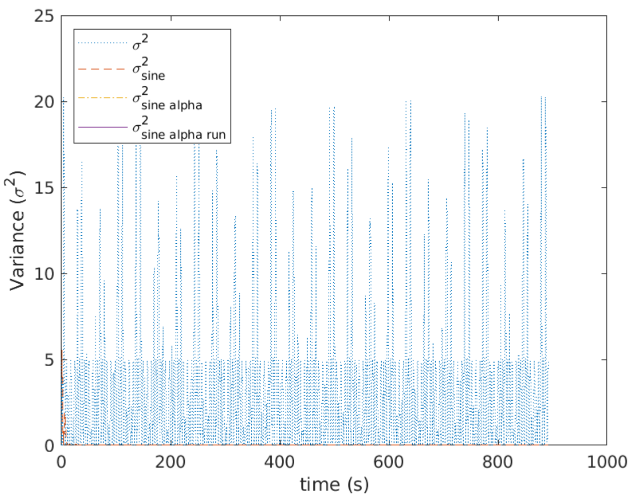

Figure 7.21: Case - A Minimum Time Sine Auxiliary Averages

7.2.4 Case F

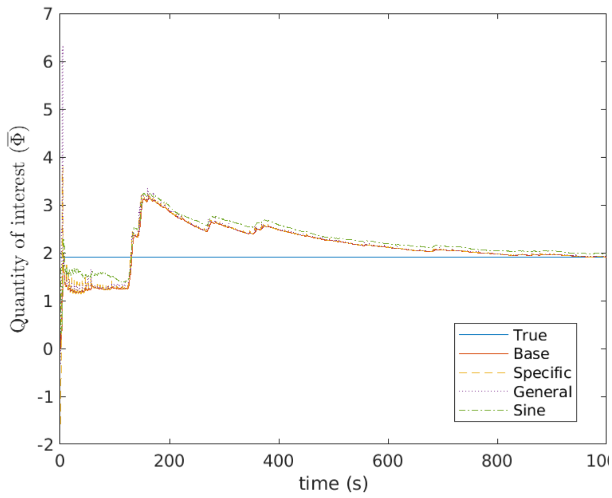

Figure 7.22: Case - C Base Averages

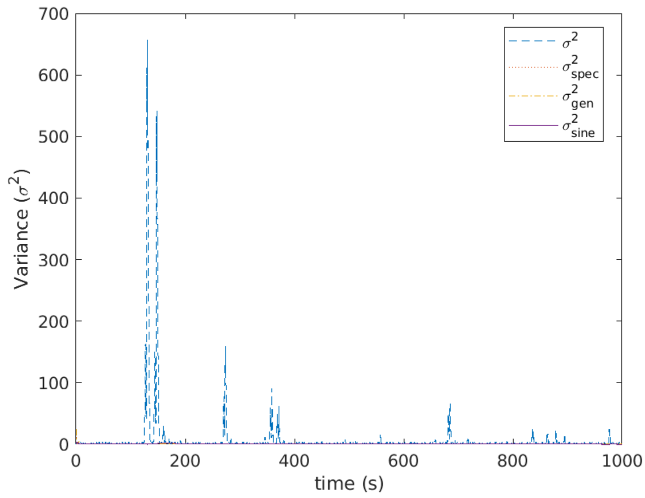

Figure 7.23: Case - C Minimum Time Base Averages

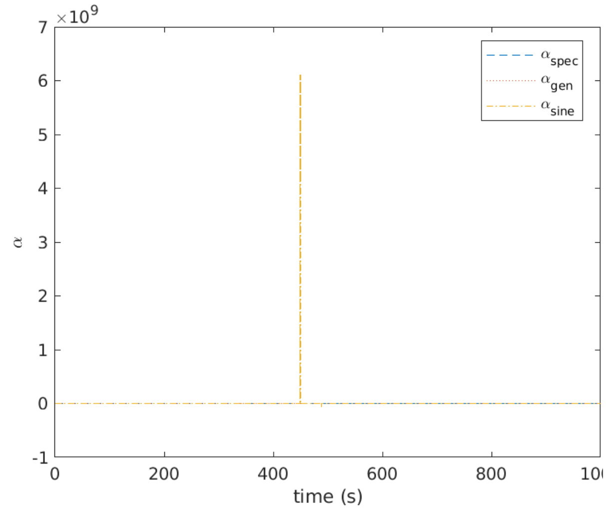

Figure 7.24: Case - C α_{min}

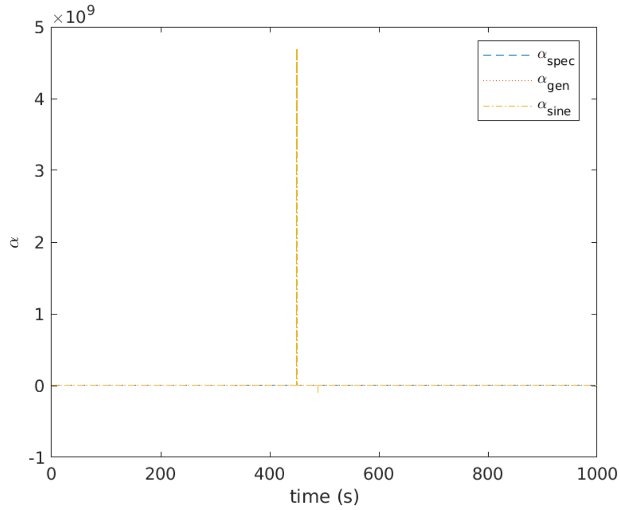

Figure 7.25: Case - C α_{run}

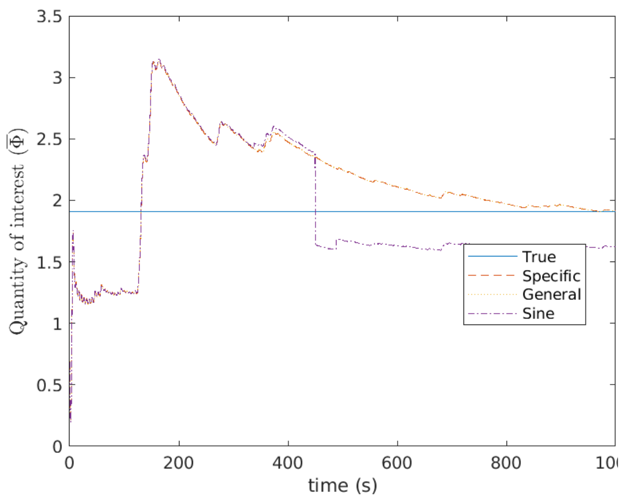

Figure 7.26: Case - C α_{min} Averages

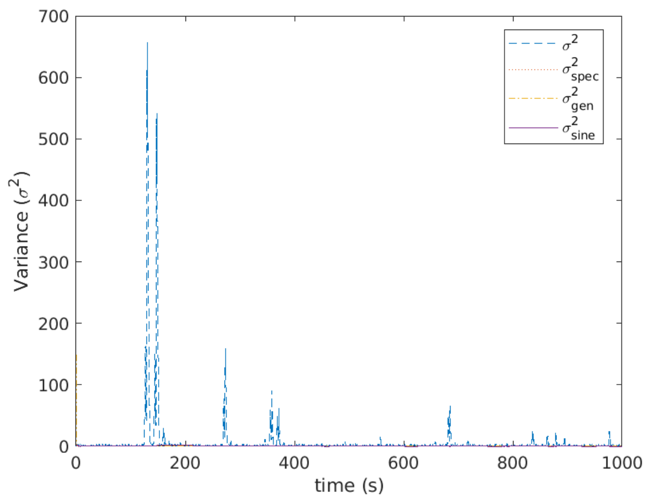

Figure 7.27: Case - C α_{min} Variances

7.3 Code

7.3.1 Initialising Code

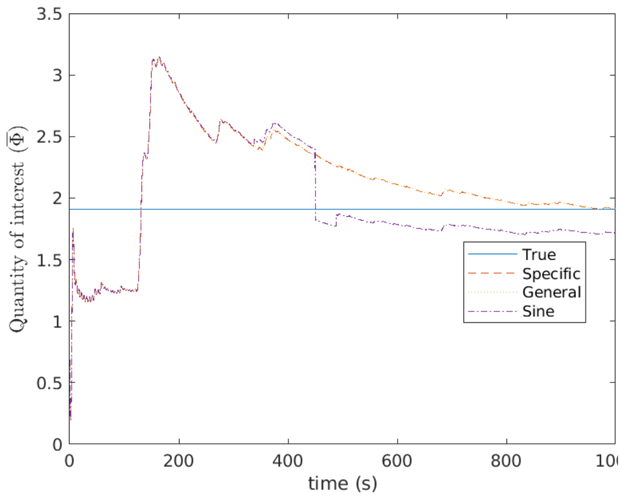

Figure 7.28: Case - C α_{run} Averages

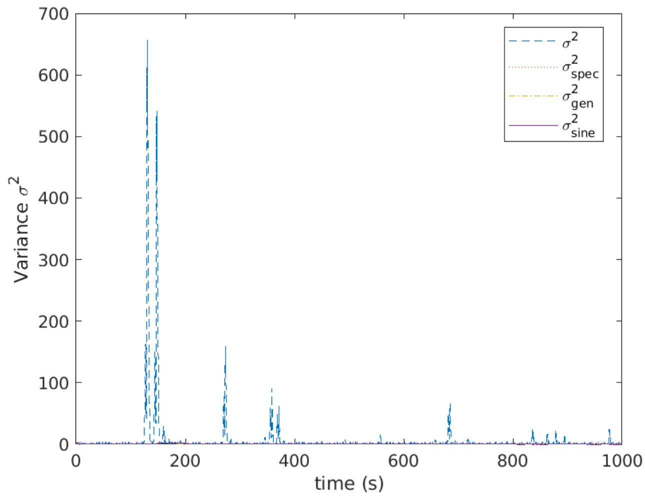

Figure 7.29: Case - C α_{run} Variances

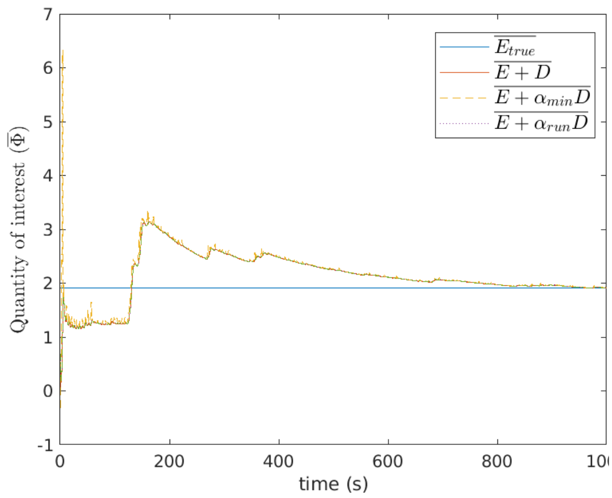

Figure 7.30: Case - C General Auxiliary Averages

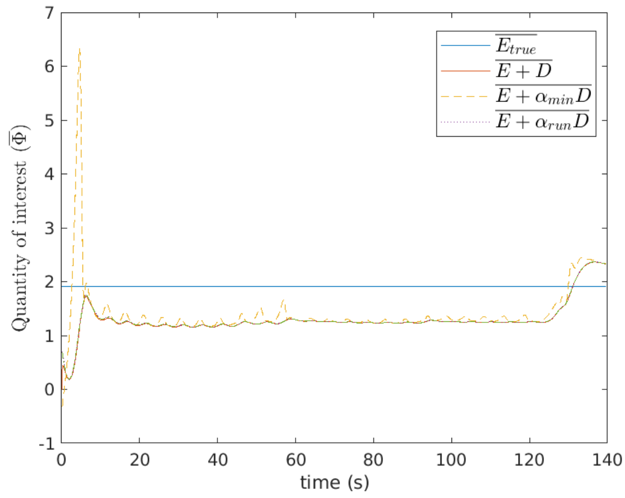

Figure 7.31: Case - C Minimum Time General Auxiliary Averages

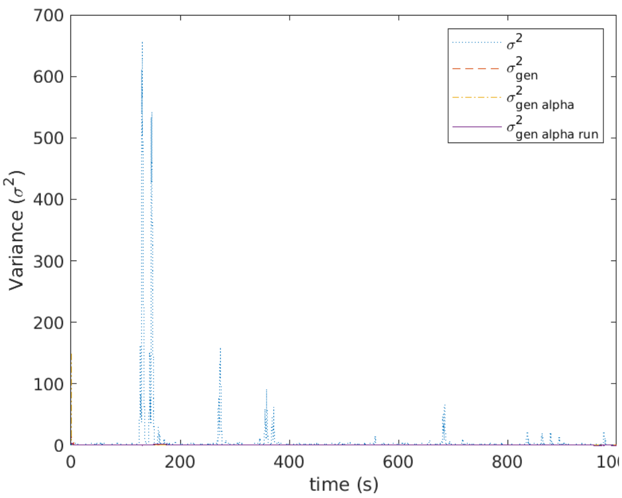

Figure 7.32: Case - C General Auxiliary Variances

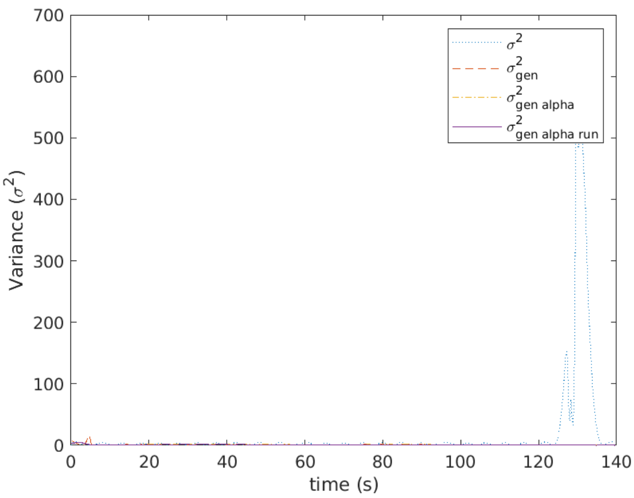

Figure 7.33: Case - C Sine Auxiliary Averages

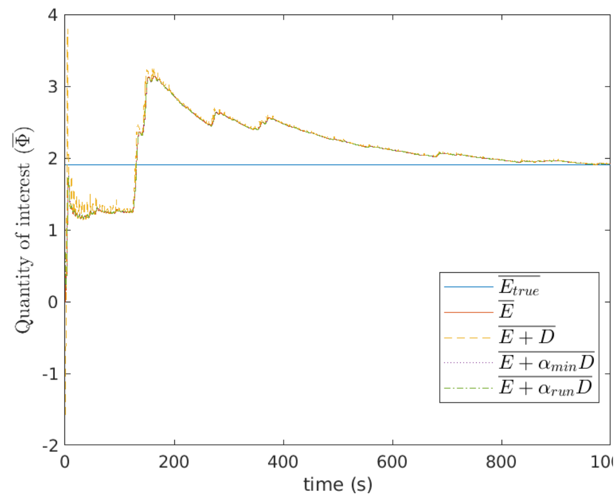

Figure 7.34: Case - C Specific Auxiliary Averages

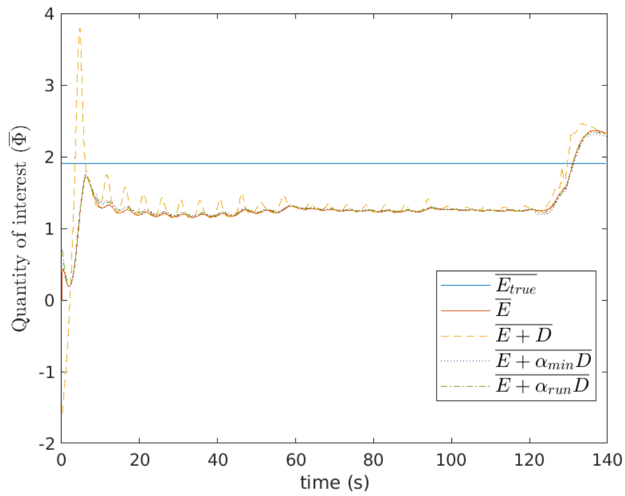

Figure 7.35: Case - C Minimum Time Specific Auxiliary Averages

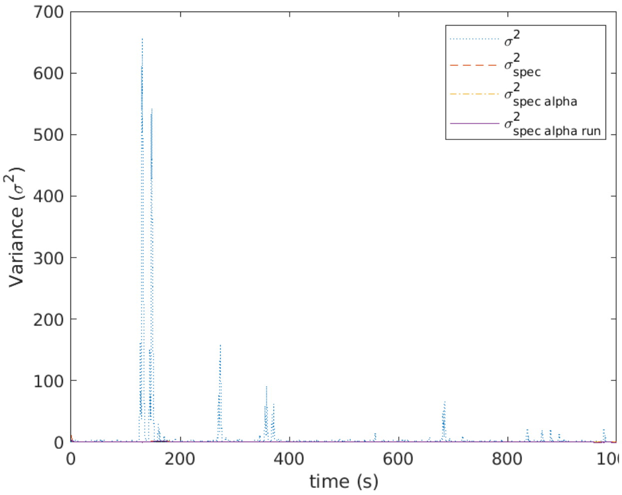

Figure 7.36: Case - C Specific Auxiliary Variances

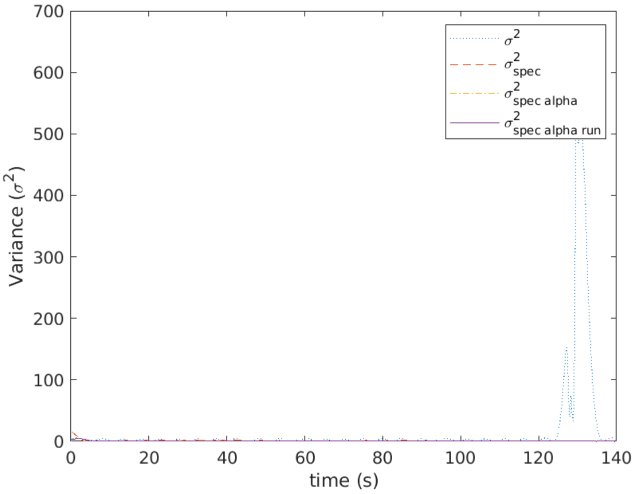

Figure 7.37: Case - C Minimum Time Specific Auxiliary Variances

```
%Function to study the initial conditions of the Rossler System for a range
%of initial conditions.
%Created by Jamell Ivan Samuels.
 clear all
 clc


%% Pre-Allocated Values

     Stored_E_bar = [];
     Stored_E_D_spec_bar = [];
     Stored_E_D_spec_alpha_bar = [];
     Stored_E_D_spec_alpha_run_bar = [];
     Stored_E_D_gen_bar = [];
     Stored_E_D_gen_alpha_bar = [];
     Stored_E_D_gen_alpha_run_bar = [];
     Stored_E_D_sine_bar = [];
     Stored_E_D_sine_alpha_bar = [];
     Stored_E_D_sine_alpha_run_bar = [];
```

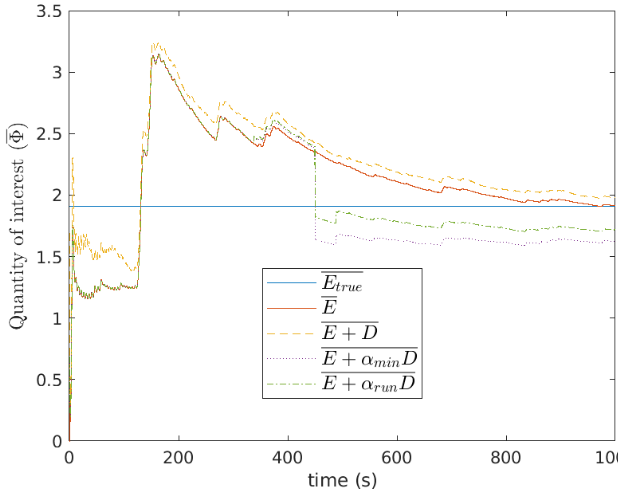

Figure 7.38: Case - C Sine Auxiliary Averages

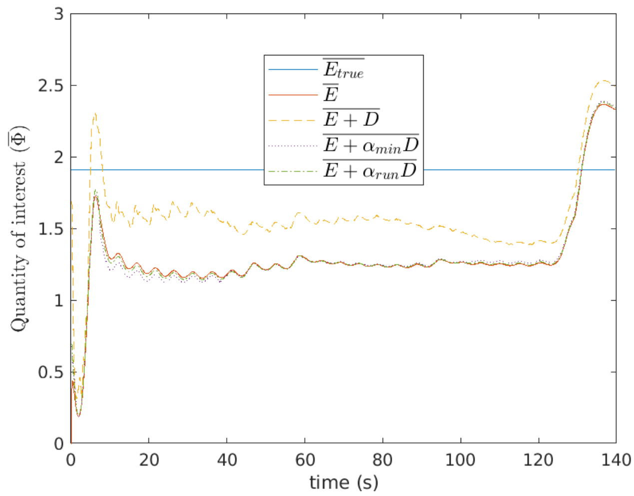

Figure 7.39: Case - C Minimum Time Sine Auxiliary Averages

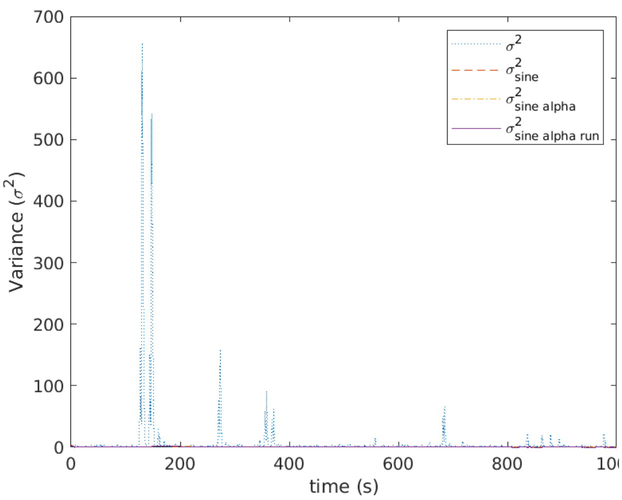

Figure 7.40: Case - C Sine Auxiliary Variances

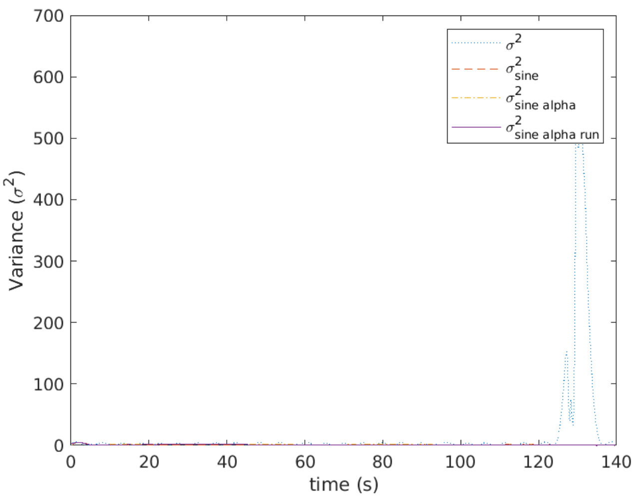

Figure 7.41: Case - C Minimum Time Sine Auxiliary Averages

```
    Stored_E_std_run_0 = [];

    %MinT Averages

    Stored_MinT_E_bar = [];
    Stored_MinT_E_D_spec_bar = [];
    Stored_MinT_E_D_spec_alpha_bar = [];
    Stored_MinT_E_D_spec_alpha_run_bar = [];
    Stored_MinT_E_D_gen_bar = [];
    Stored_MinT_E_D_gen_alpha_bar = [];
    Stored_MinT_E_D_gen_alpha_run_bar = [];
    Stored_MinT_E_D_sine_bar = [];
    Stored_MinT_E_D_sine_alpha_bar = [];
    Stored_MinT_E_D_sine_alpha_run_bar= [];

    %Variances

    Stored_Var_bar = [];
    Stored_Var_spec_bar = [];
```

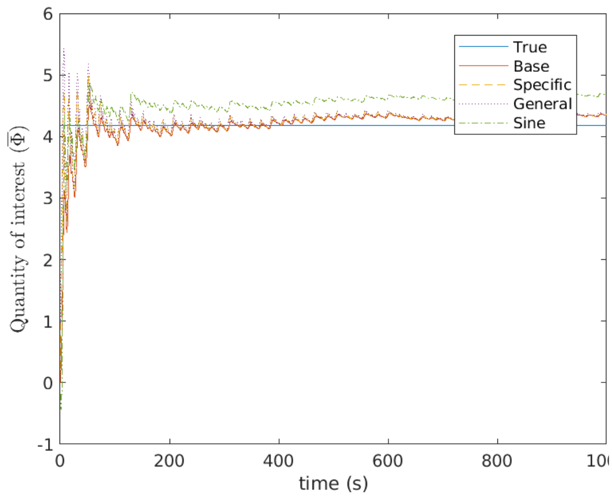

Figure 7.42: Case - E Base Averages

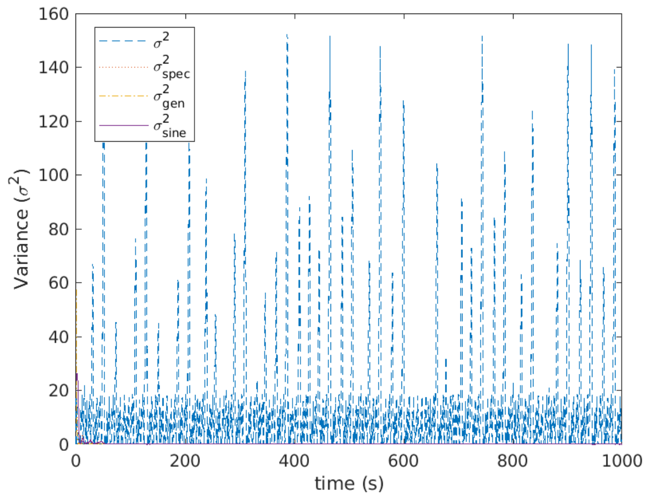

Figure 7.43: Case - E Minimum Time Base Averages

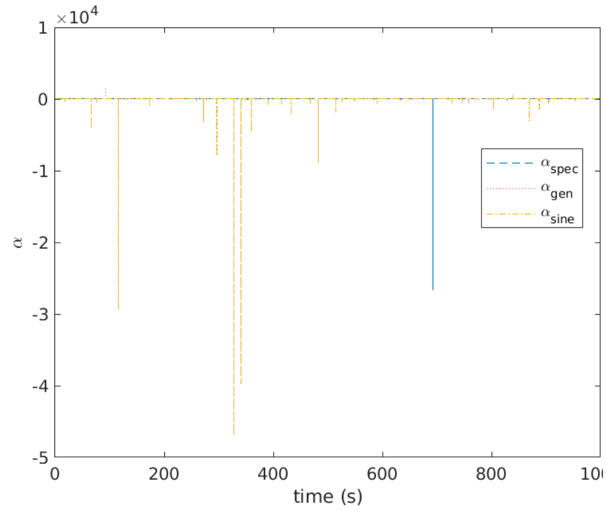

Figure 7.44: Case - E α_{min}

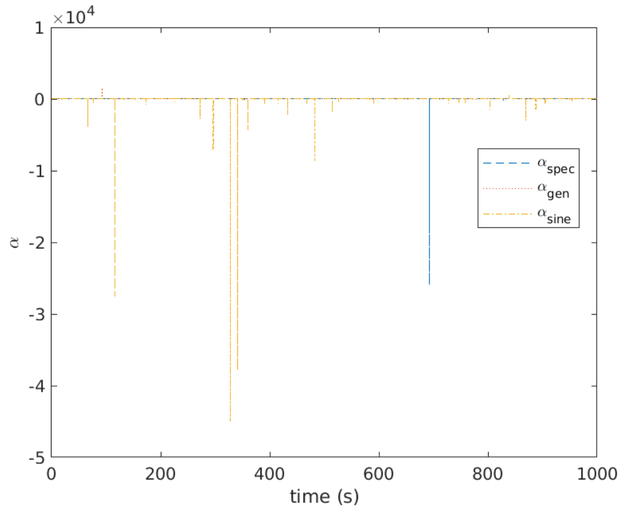

Figure 7.45: Case - E α_{run}

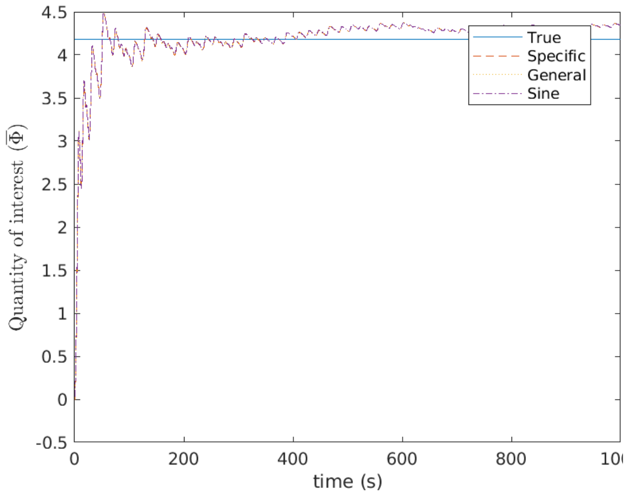

Figure 7.46: Case - E α_{min} Averages

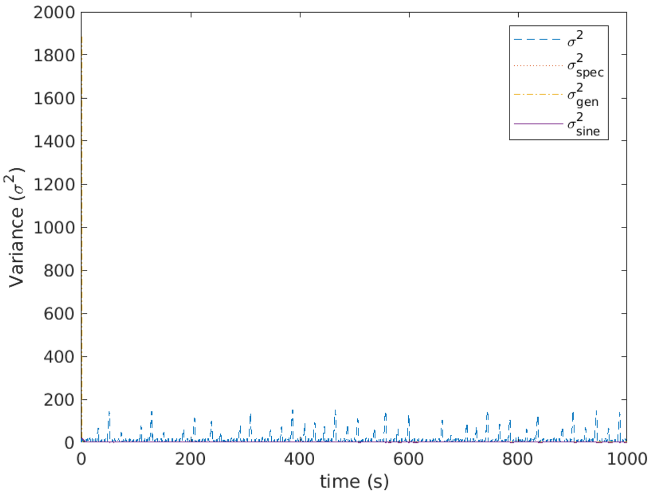

Figure 7.47: Case - E α_{min} Variances

```
    Stored_Var_alpha_spec_bar = [];
    Stored_Var_alpha_run_spec_bar = [];
```

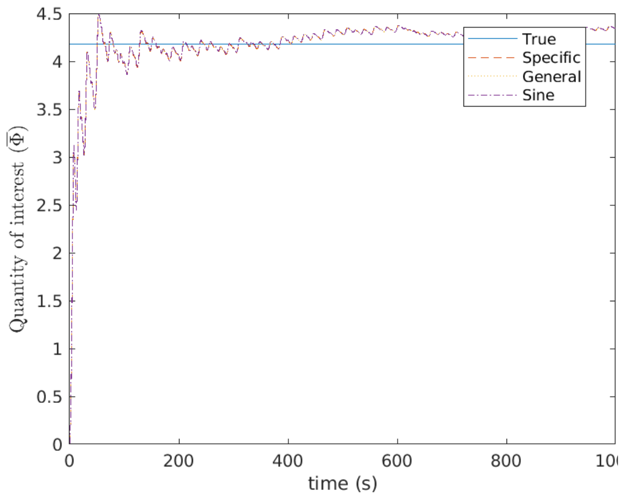

Figure 7.48: Case - E α_{run} Averages

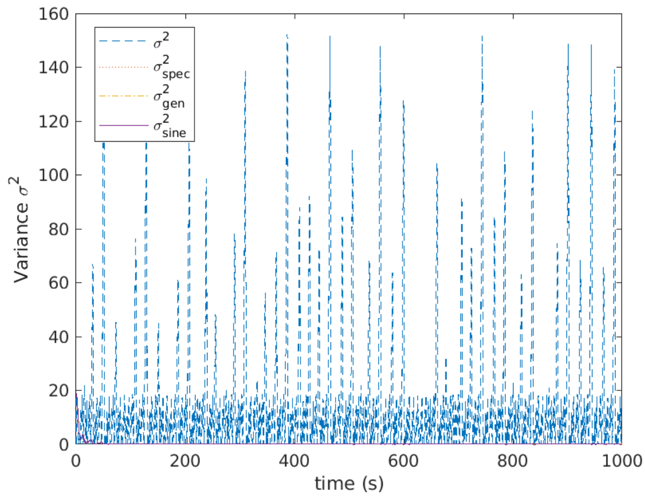

Figure 7.49: Case - E α_{run} Variances

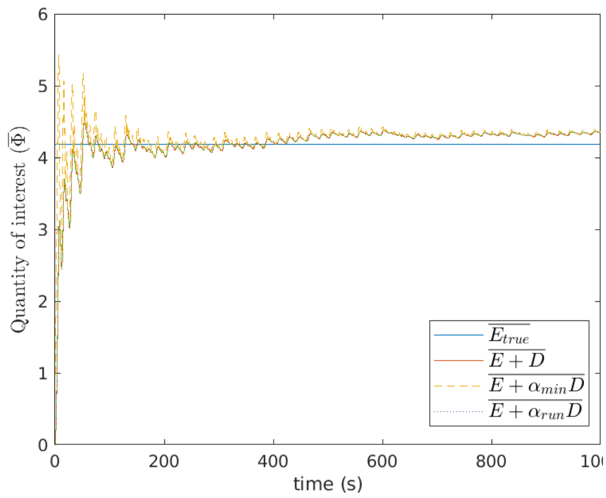

Figure 7.50: Case - E General Auxiliary Averages

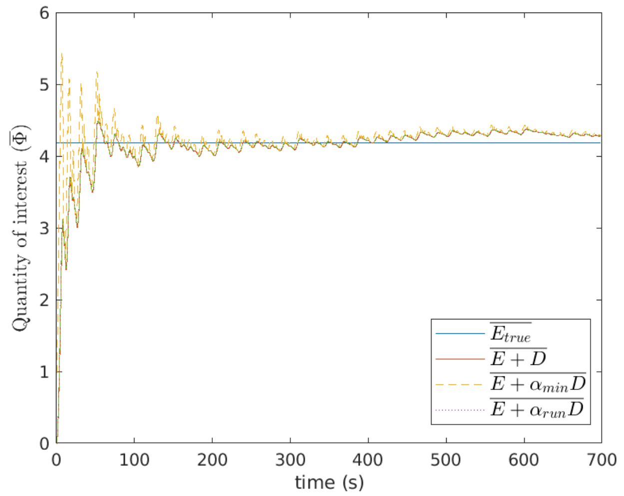

Figure 7.51: Case - E Minimum Time General Auxiliary Averages

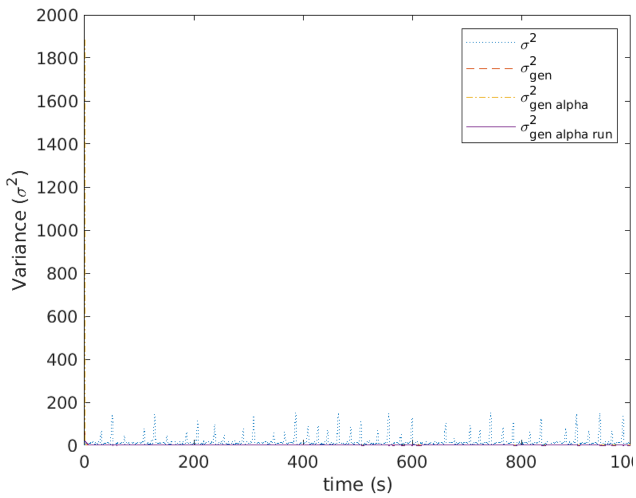

Figure 7.52: Case - E General Auxiliary Variances

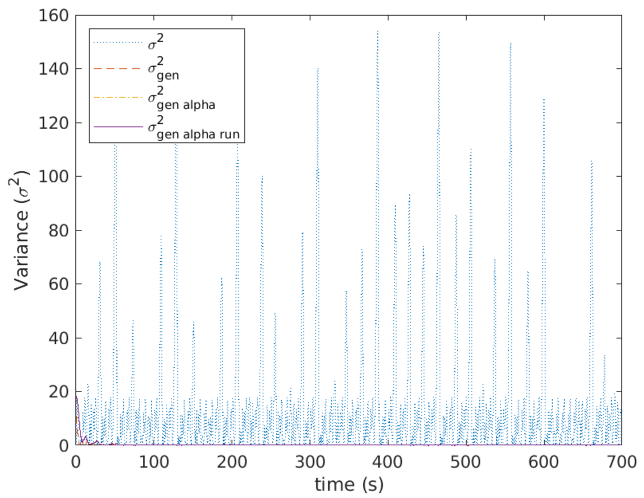

Figure 7.53: Case - E Sine Auxiliary Averages

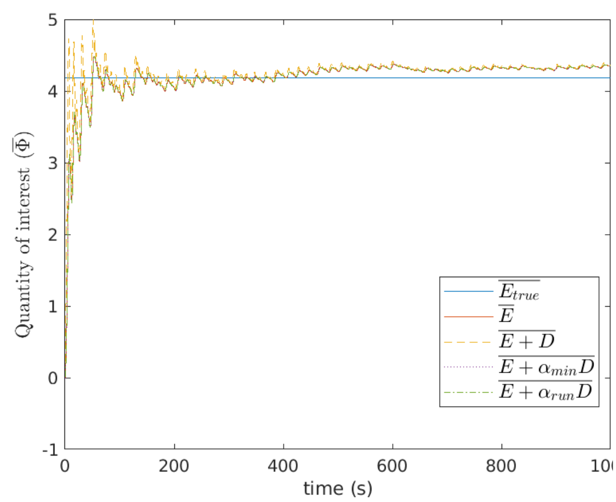

Figure 7.54: Case - E Specific Auxiliary Averages

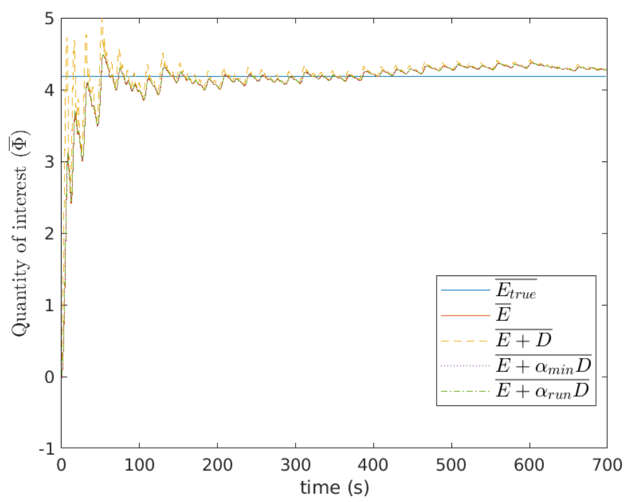

Figure 7.55: Case - E Minimum Time Specific Auxiliary Averages

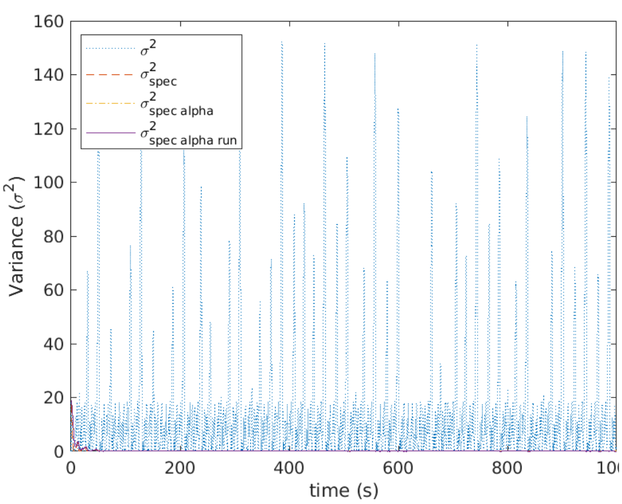

Figure 7.56: Case - E Specific Auxiliary Variances

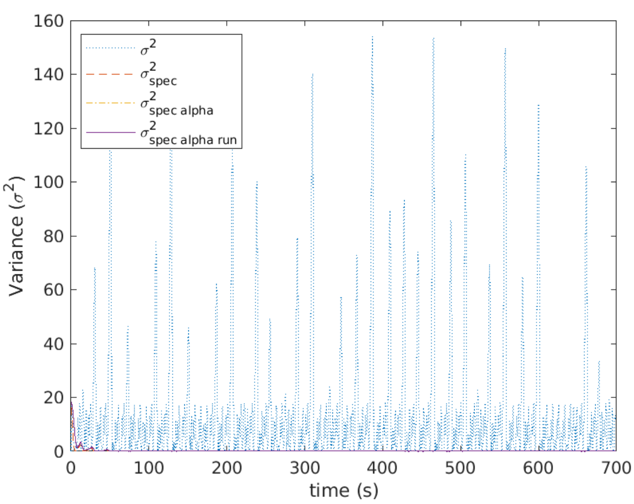

Figure 7.57: Case - E Minimum Time Specific Auxiliary Variances

```
    Stored_Var_gen_bar = [];
    Stored_Var_alpha_gen_bar = [];
    Stored_Var_alpha_run_gen_bar = [];
    Stored_Var_sine_bar = [];
    Stored_Var_alpha_sine_bar = [];
    Stored_Var_alpha_run_sine_bar = [];

    %MinT Variances

    Stored_MinT_Var_bar = [];
    Stored_MinT_Var_spec_bar = [];
    Stored_MinT_Var_alpha_spec_bar = [];
    Stored_MinT_Var_alpha_run_spec_bar = [];
    Stored_MinT_Var_gen_bar = [];
    Stored_MinT_Var_alpha_gen_bar = [];
    Stored_MinT_Var_alpha_run_gen_bar = [];
    Stored_MinT_Var_sine_bar = [];
    Stored_MinT_Var_alpha_sine_bar = [];
    Stored_MinT_Var_alpha_run_sine_bar = [];
```

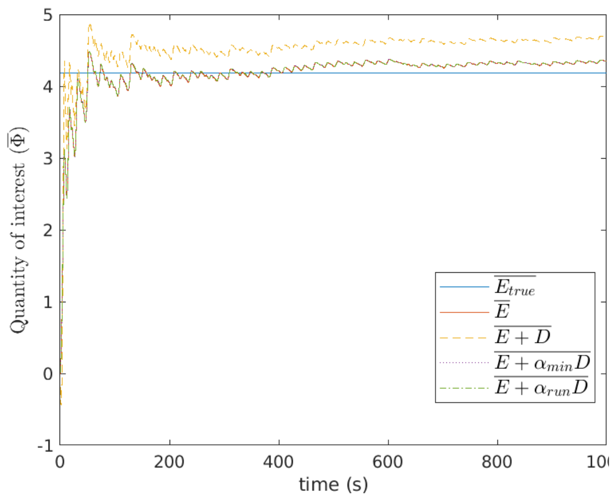

Figure 7.58: Case - E Sine Auxiliary Averages

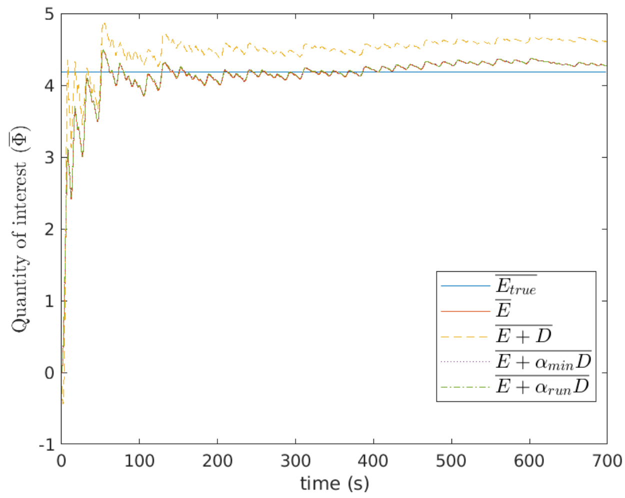

Figure 7.59: Case - E Minimum Time Sine Auxiliary Averages

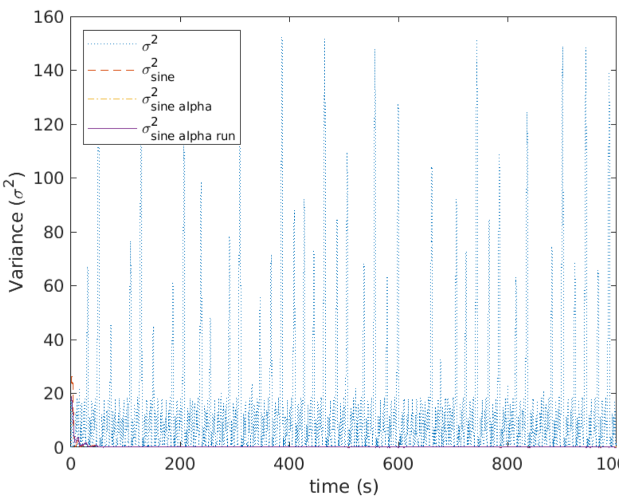

Figure 7.60: Case - E Sine Auxiliary Variances

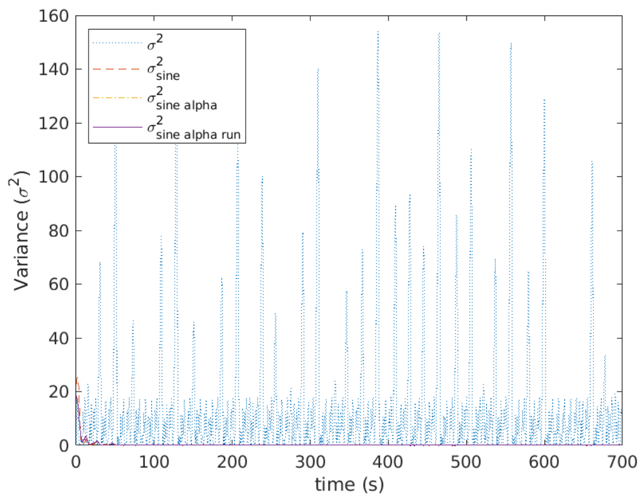

Figure 7.61: Case - E Minimum Time Sine Auxiliary Averages

```
    Stored_x0 = [];


%% Initial Conditions
    for p = 1:5;

    clearvars -except Stored_x0 Stored_E_bar Stored_E_D_spec_bar
        Stored_E_D_spec_alpha_bar Stored_E_D_gen_bar ...
    Stored_E_D_gen_alpha_bar Stored_E_D_gen_alpha_run_bar Stored_E_D_sine_bar
        Stored_E_D_sine_alpha_bar ...
    Stored_E_D_sine_alpha_run_bar Stored_E_std_run_0 Stored_MinT_E_bar
        Stored_MinT_E_D_spec_bar ...
    Stored_MinT_E_D_spec_alpha_bar Stored_MinT_E_D_spec_alpha_run_bar
        Stored_MinT_E_D_gen_bar Stored_MinT_E_D_gen_alpha_bar ...
    Stored_MinT_E_D_gen_alpha_run_bar Stored_MinT_E_D_sine_bar
        Stored_MinT_E_D_sine_alpha_bar Stored_MinT_E_D_sine_alpha_run_bar...
    Stored_Var_bar Stored_Var_spec_bar Stored_Var_alpha_spec_bar
        Stored_Var_alpha_run_spec_bar Stored_Var_gen_bar ...
    Stored_Var_alpha_gen_bar Stored_Var_alpha_run_gen_bar Stored_Var_sine_bar
```

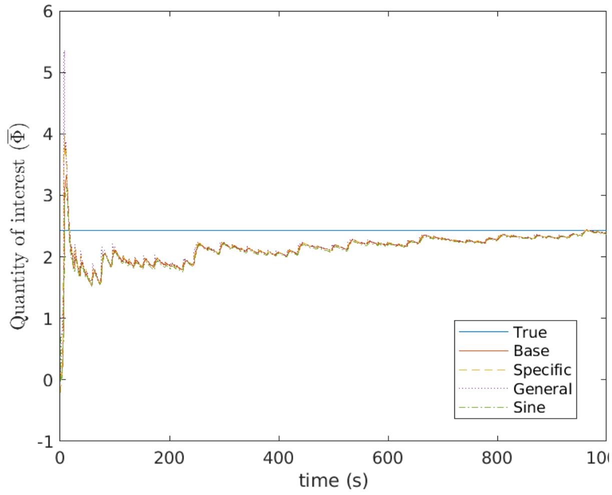

Figure 7.62: Case - F Base Averages

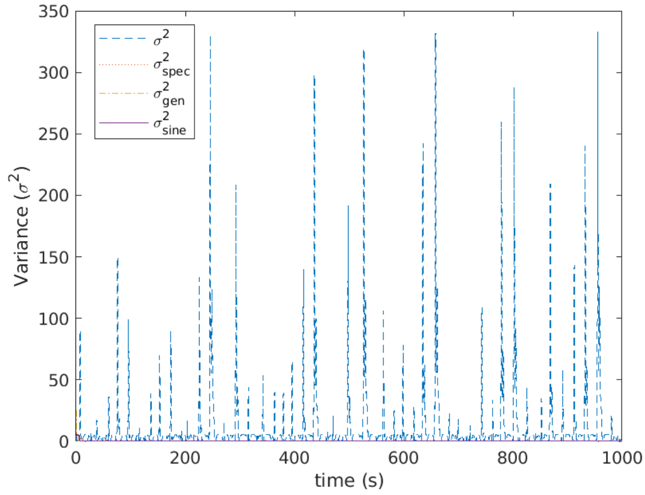

Figure 7.63: Case - F Minimum Time Base Averages

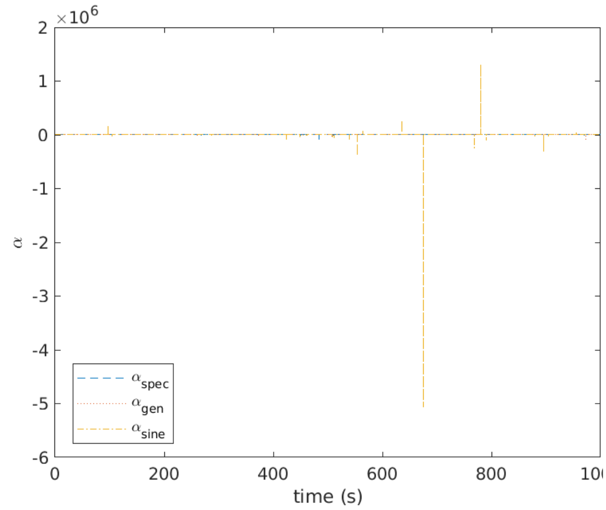

Figure 7.64: Case - F α_{min}

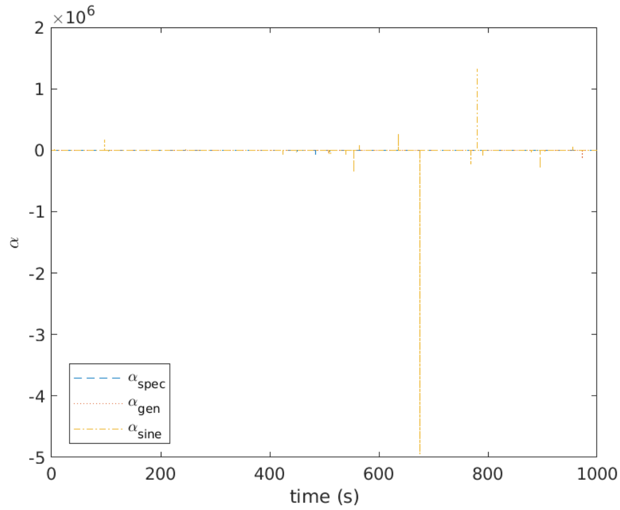

Figure 7.65: Case - F α_{run}

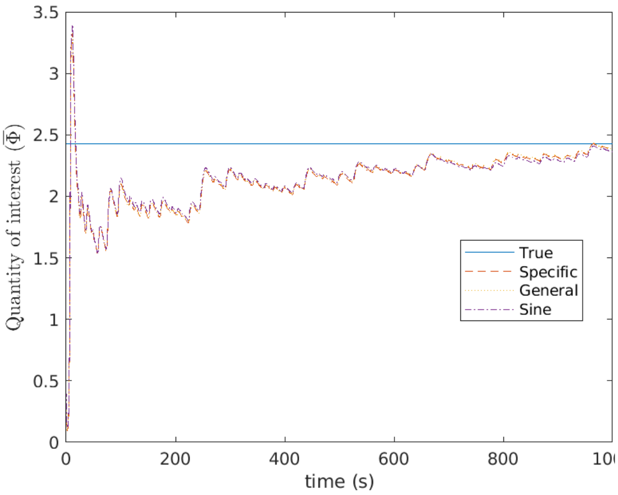

Figure 7.66: Case - F α_{min} Averages

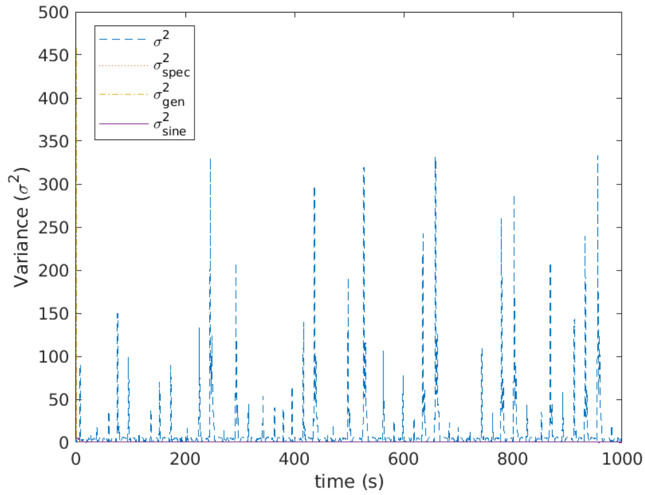

Figure 7.67: Case - F α_{min} Variances

```
    Stored_Var_alpha_sine_bar  Stored_Var_alpha_run_sine_bar ...
Stored_Var_bar  Stored_Var_spec_bar  Stored_Var_alpha_spec_bar
```

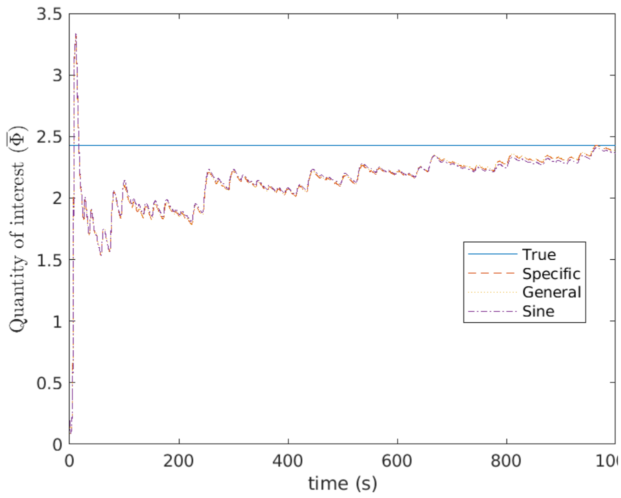

Figure 7.68: Case - F α_{run} Averages

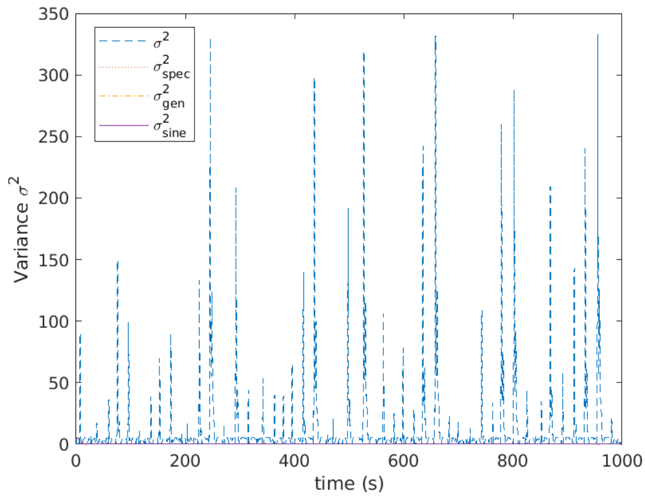

Figure 7.69: Case - F α_{run} Variances

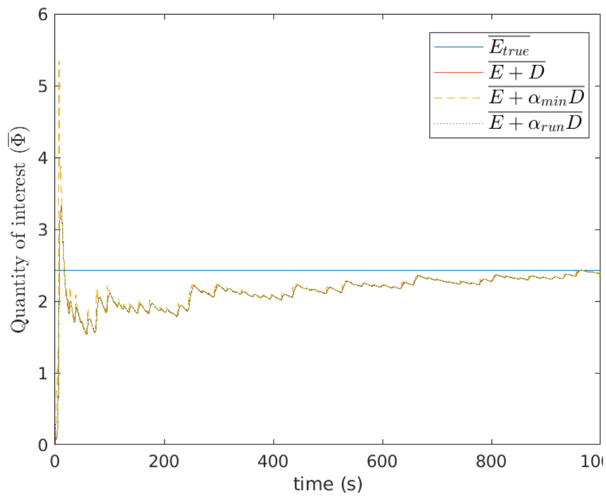

Figure 7.70: Case - F General Auxiliary Averages

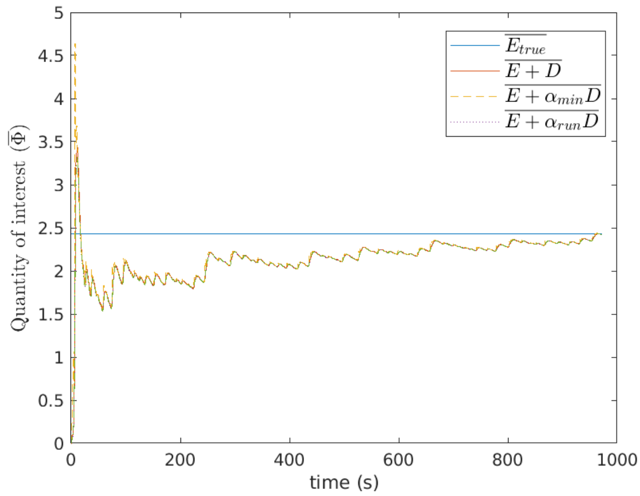

Figure 7.71: Case - F Minimum Time General Auxiliary Averages

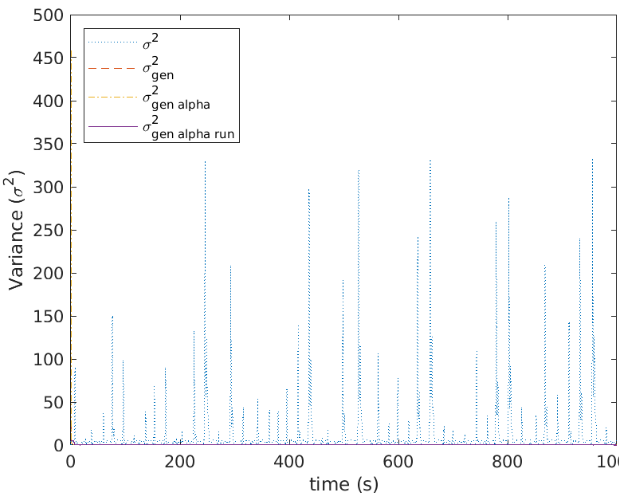

Figure 7.72: Case - F General Auxiliary Variances

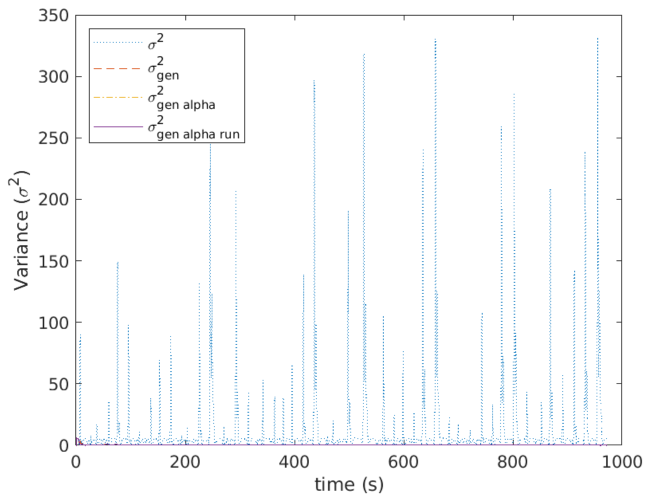

Figure 7.73: Case - F Sine Auxiliary Averages

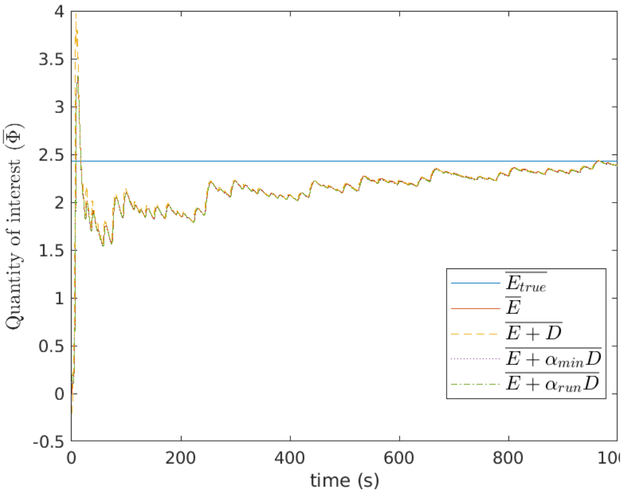

Figure 7.74: Case - F Specific Auxiliary Averages

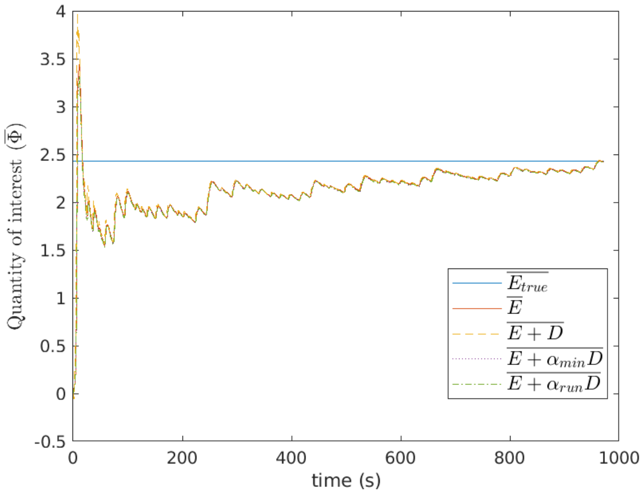

Figure 7.75: Case - F Minimum Time Specific Auxiliary Averages

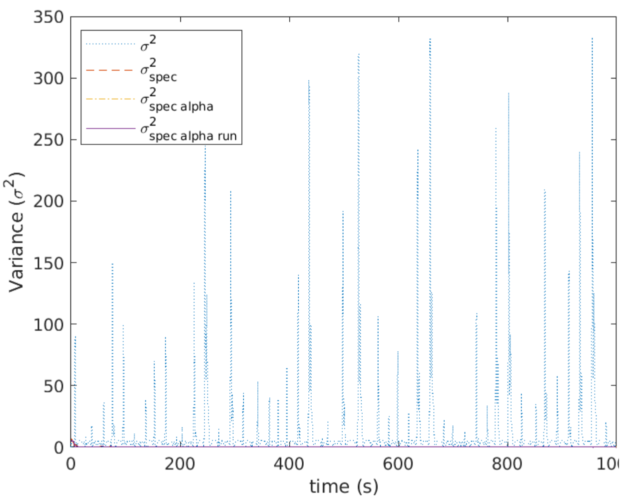

Figure 7.76: Case - F Specific Auxiliary Variances

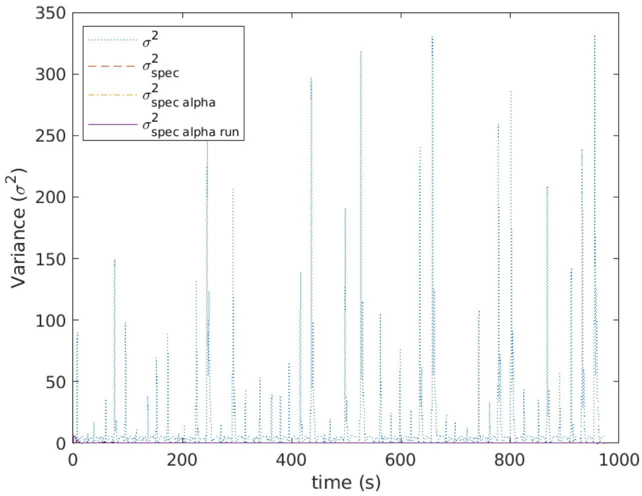

Figure 7.77: Case - F Minimum Time Specific Auxiliary Variances

```
        Stored_Var_alpha_run_spec_bar  Stored_Var_gen_bar ...
    Stored_Var_alpha_gen_bar  Stored_Var_alpha_run_gen_bar  Stored_Var_sine_bar
        Stored_Var_alpha_sine_bar  Stored_Var_alpha_run_sine_bar ...
    Stored_MinT_Var_bar  Stored_MinT_Var_spec_bar  Stored_MinT_Var_alpha_spec_bar
        Stored_MinT_Var_alpha_run_spec_bar ...
    Stored_MinT_Var_gen_bar  Stored_MinT_Var_alpha_gen_bar
        Stored_MinT_Var_alpha_run_gen_bar  Stored_MinT_Var_sine_bar  ...
    Stored_MinT_Var_alpha_sine_bar  Stored_MinT_Var_alpha_run_sine_bar  p
    disp('Current System')
    disp(p)

    delta_t = 0.1;
    T = 1000;

    if p == 1; %Case A
    x0 = [0.014,0,-0.014];
    F = @(t,x) [x(2);-x(1)+x(2)*x(3);1-x(2)^2];
    tspan = [0:delta_t:T];
    eps = 0.0000000001;
```

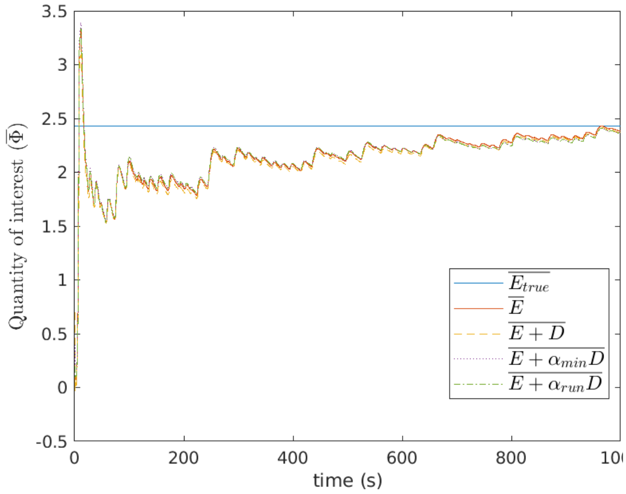

Figure 7.78: Case - F Sine Auxiliary Averages

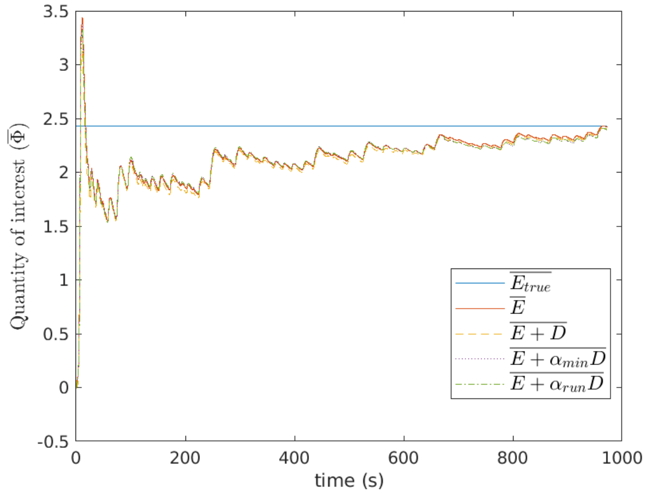

Figure 7.79: Case - F Minimum Time Sine Auxiliary Averages

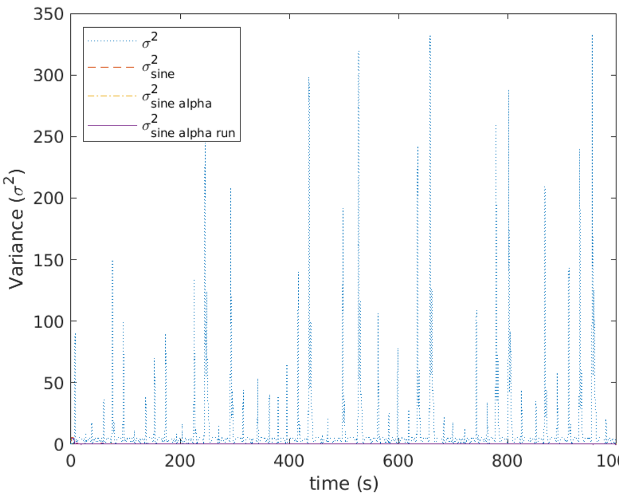

Figure 7.80: Case - F Sine Auxiliary Variances

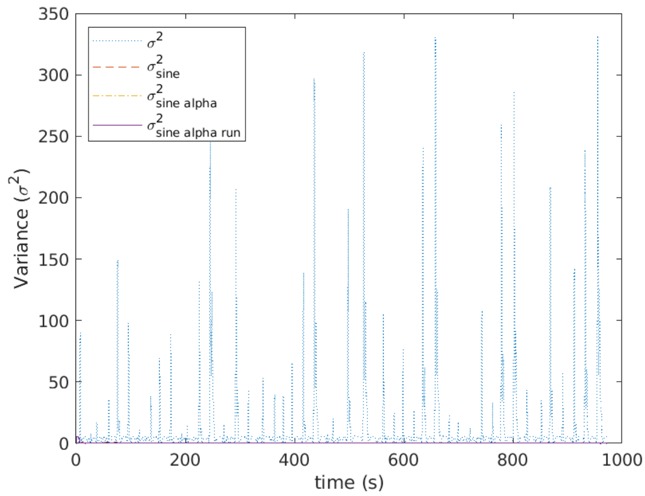

Figure 7.81: Case - F Minimum Time Sine Auxiliary Averages

```
    options = odeset('RelTol',eps,'AbsTol',[eps eps eps/20]);
    [t,x] = ode45(F,tspan,x0,options);
    T = t(end);
    disp('Time actually ran for')
    disp(T)

    elseif p == 2; %Case B

    x0 = [0.210,0,-1.20];
    F = @(t,x) [x(2)*x(3);x(1)-x(2);1-x(1)*x(2)];
    tspan = [0:delta_t:T];
    eps = 0.0000000001;
    options = odeset('RelTol',eps,'AbsTol',[eps eps eps/20]);
    [t,x] = ode45(F,tspan,x0,options);
    T = t(end);
    disp('Time actually ran for')
    disp(T)
```

```
elseif p == 3; %Case C

x0=[0.163,0,-1.163];
F = @(t,x) [x(2)*x(3);x(1)-x(2);1-x(1)^2];
tspan = [0:delta_t:T];
eps = 0.0000000001;
options = odeset('RelTol',eps,'AbsTol',[eps eps eps/20]);
[t,x] = ode45(F,tspan,x0,options);
T = t(end);
disp('Time actually ran for')
disp(T)

elseif p == 4; %Case F

x0 = [0.117,0,-0.617];
F = @(t,x) [x(2)+x(3);-x(1)+0.5*x(2);x(1)^2-x(3)];
tspan = [0:delta_t:T];
eps = 0.0000000001;
options = odeset('RelTol',eps,'AbsTol',[eps eps eps/20]);
[t,x] = ode45(F,tspan,x0,options);
T = t(end);
disp('Time actually ran for')
disp(T)


elseif p == 5; %Case E

x0=[0.078,0,0.117]
F = @(t,x) [x(2)*x(3); x(1)^2-x(2); 1-4*x(1)];
tspan = [0:delta_t:T];
eps = 0.0000000001;
options = odeset('RelTol',eps,'AbsTol',[eps eps eps/20]);
[t,x] = ode45(F,tspan,x0,options);
T = t(end);
disp('Time actually ran for')
disp(T)
end



System_Base_sprott;



%Averages
Stored_x0(p,:) = x0;
Stored_E_bar(p) = E_bar;
Stored_E_D_spec_bar(p) = E_D_spec_bar;
Stored_E_D_spec_alpha_bar(p) = E_D_spec_alpha_bar;
Stored_E_D_spec_alpha_run_bar(p) = E_D_spec_alpha_run_bar;
Stored_E_D_gen_bar(p) = E_D_gen_bar;
Stored_E_D_gen_alpha_bar(p) = E_D_gen_alpha_bar;
Stored_E_D_gen_alpha_run_bar(p) =  E_D_gen_alpha_run_bar;
Stored_E_D_sine_bar(p) = E_D_sine_bar;
Stored_E_D_sine_alpha_bar(p) = E_D_sine_alpha_bar;
Stored_E_D_sine_alpha_run_bar(p) = E_D_sine_alpha_run_bar;
Stored_E_std_run_0(p) = E_std_run_0;

%MinT Averages

```

```
    Stored_MinT_E_bar(p) = MinT_E_bar;
    Stored_MinT_E_D_spec_bar(p) = MinT_E_D_spec_bar;
    Stored_MinT_E_D_spec_alpha_bar(p) = MinT_E_D_spec_alpha_bar;
    Stored_MinT_E_D_spec_alpha_run_bar(p) = MinT_E_D_spec_alpha_run_bar;
    Stored_MinT_E_D_gen_bar(p) = MinT_E_D_gen_bar;
    Stored_MinT_E_D_gen_alpha_bar(p) = MinT_E_D_gen_alpha_bar;
    Stored_MinT_E_D_gen_alpha_run_bar(p) = MinT_E_D_gen_alpha_run_bar;
    Stored_MinT_E_D_sine_bar(p) = MinT_E_D_sine_bar;
    Stored_MinT_E_D_sine_alpha_bar(p) = MinT_E_D_sine_alpha_bar;
    Stored_MinT_E_D_sine_alpha_run_bar(p) = MinT_E_D_sine_alpha_run_bar;

    %Variances

    Stored_Var_bar(p) = Var_bar;
    Stored_Var_spec_bar(p) = Var_spec_bar;
    Stored_Var_alpha_spec_bar(p) = Var_alpha_spec_bar;
    Stored_Var_alpha_run_spec_bar(p) = Var_alpha_run_spec_bar;
    Stored_Var_gen_bar(p) = Var_gen_bar;
    Stored_Var_alpha_gen_bar(p) = Var_alpha_gen_bar;
    Stored_Var_alpha_run_gen_bar(p) = Var_alpha_run_gen_bar;
    Stored_Var_sine_bar(p) = Var_alpha_sine_bar;
    Stored_Var_alpha_sine_bar(p) = Var_alpha_sine_bar;
    Stored_Var_alpha_run_sine_bar(p) = Var_alpha_run_sine_bar;

    %MinT Variances

    Stored_MinT_Var_bar(p) = MinT_Var_bar;
    Stored_MinT_Var_spec_bar(p) = MinT_Var_spec_bar;
    Stored_MinT_Var_alpha_spec_bar(p) = MinT_Var_alpha_spec_bar;
    Stored_MinT_Var_alpha_run_spec_bar(p) = MinT_Var_alpha_run_spec_bar;
    Stored_MinT_Var_gen_bar(p) = MinT_Var_gen_bar;
    Stored_MinT_Var_alpha_gen_bar(p) = MinT_Var_alpha_gen_bar;
    Stored_MinT_Var_alpha_run_gen_bar(p) = MinT_Var_alpha_run_gen_bar;
    Stored_MinT_Var_sine_bar(p) = MinT_Var_sine_bar;
    Stored_MinT_Var_alpha_sine_bar(p) = MinT_Var_alpha_sine_bar;
    Stored_MinT_Var_alpha_run_sine_bar(p) = MinT_Var_alpha_run_sine_bar;

    %Variables you should know
    Stored_T(p) = T;
    Stored_MinT(p) = minT;
    Stored_DD_bar_spec(p) = DD_bar_spec;
    Stored_DD_bar_gen(p) = DD_bar_gen;
    Stored_DD_bar_sine(p) = DD_bar_sine;
    Stored_MinT_DD_bar_spec(p)= MinT_DD_bar_spec;
    Stored_MinT_DD_bar_gen(p) = MinT_DD_bar_gen;
    Stored_MinT_DD_bar_sine(p) = MinT_DD_bar_sine;




    Stored_DD_bar_spec = nonzeros(Stored_DD_bar_spec);
    Stored_DD_bar_gen = nonzeros(Stored_DD_bar_gen);
    Stored_DD_bar_sine = nonzeros(Stored_DD_bar_sine);
    Stored_MinT_DD_bar_spec = nonzeros(Stored_MinT_DD_bar_spec);
    Stored_MinT_DD_bar_gen = nonzeros(Stored_MinT_DD_bar_gen);
    Stored_MinT_DD_bar_sine = nonzeros(Stored_MinT_DD_bar_sine);
    Stored_T = nonzeros(Stored_T);
    Stored_MinT = nonzeros(Stored_MinT);
    Stored_x0 = Stored_x0(any(Stored_x0,2),:);  %rows
    Stored_E_bar = nonzeros(Stored_E_bar);
```

```
    Stored_E_D_spec_bar = nonzeros(Stored_E_bar);
    Stored_E_D_spec_alpha_bar = nonzeros(Stored_E_bar);
    Stored_E_D_gen_bar = nonzeros(Stored_E_D_gen_bar) ;
    Stored_E_D_gen_alpha_bar = nonzeros(Stored_E_D_gen_alpha_bar);
    Stored_E_D_gen_alpha_run_bar = nonzeros(Stored_E_D_gen_alpha_run_bar);
    Stored_E_D_sine_bar = nonzeros(Stored_E_D_sine_bar);
    Stored_E_D_sine_alpha_bar = nonzeros(Stored_E_D_sine_alpha_bar);
    Stored_E_D_sine_alpha_run_bar = nonzeros(Stored_E_D_sine_alpha_run_bar);
    Stored_E_std_run_0 = nonzeros(Stored_E_std_run_0);


   %MinT Averages

    Stored_MinT_E_bar = nonzeros(Stored_MinT_E_bar);
    Stored_MinT_E_D_spec_bar = nonzeros(Stored_MinT_E_D_spec_bar);
    Stored_MinT_E_D_spec_alpha_bar = nonzeros(Stored_MinT_E_D_spec_alpha_bar);
    Stored_MinT_E_D_spec_alpha_run_bar = nonzeros(
        Stored_MinT_E_D_spec_alpha_run_bar);
    Stored_MinT_E_D_gen_bar = nonzeros(Stored_MinT_E_D_gen_bar);
    Stored_MinT_E_D_gen_alpha_bar = nonzeros(Stored_MinT_E_D_gen_alpha_bar);
    Stored_MinT_E_D_gen_alpha_run_bar = nonzeros(
        Stored_MinT_E_D_gen_alpha_run_bar);
    Stored_MinT_E_D_sine_bar = nonzeros(Stored_MinT_E_D_sine_bar);
    Stored_MinT_E_D_sine_alpha_bar = nonzeros(Stored_MinT_E_D_sine_alpha_bar);
    Stored_MinT_E_D_sine_alpha_run_bar = nonzeros(
        Stored_MinT_E_D_sine_alpha_run_bar);

   %Variances

    Stored_Var_bar = nonzeros(Stored_Var_bar);
    Stored_Var_spec_bar = nonzeros(Stored_Var_spec_bar);
    Stored_Var_alpha_spec_bar = nonzeros(Stored_Var_alpha_spec_bar);
    Stored_Var_alpha_run_spec_bar = nonzeros(Stored_Var_alpha_run_spec_bar);
    Stored_Var_gen_bar = nonzeros(Stored_Var_gen_bar);
    Stored_Var_alpha_gen_bar = nonzeros(Stored_Var_alpha_gen_bar);
    Stored_Var_alpha_run_gen_bar = nonzeros(Stored_Var_alpha_run_gen_bar);
    Stored_Var_sine_bar = nonzeros(Stored_Var_sine_bar);
    Stored_Var_alpha_sine_bar = nonzeros(Stored_Var_alpha_sine_bar);
    Stored_Var_alpha_run_sine_bar = nonzeros(Stored_Var_alpha_run_sine_bar);

   %MinT Variances

    Stored_MinT_Var_bar = nonzeros(Stored_MinT_Var_bar);
    Stored_MinT_Var_spec_bar = nonzeros(Stored_MinT_Var_spec_bar);
    Stored_MinT_Var_alpha_spec_bar = nonzeros(Stored_MinT_Var_alpha_spec_bar);
    Stored_MinT_Var_alpha_run_spec_bar = nonzeros(
        Stored_MinT_Var_alpha_run_spec_bar);
    Stored_MinT_Var_gen_bar = nonzeros(Stored_MinT_Var_gen_bar);
    Stored_MinT_Var_alpha_gen_bar = nonzeros(Stored_MinT_Var_alpha_gen_bar);
    Stored_MinT_Var_alpha_run_gen_bar = nonzeros(
        Stored_MinT_Var_alpha_run_gen_bar);
    Stored_MinT_Var_sine_bar = nonzeros(Stored_MinT_Var_sine_bar);
    Stored_MinT_Var_alpha_sine_bar = nonzeros(Stored_MinT_Var_alpha_sine_bar);
    Stored_MinT_Var_alpha_run_sine_bar = nonzeros(
        Stored_MinT_Var_alpha_run_sine_bar);
    end

   %errors
    E_infty = 2.2
    E_bar_error = (E_bar-
```

```
plot_function

MinT_plot_function

save('All')
```

7.3.2 Base Code

```
% Rossler System - Base Framework
%Created by Jamell Ivan Samuels


%% Initial Conditions

%delta_t = 0.1;
%T = 1000;
%x0 = [3.219767,17.917641,-9.154723]; %Case A
%a = 0.1;
%b = 0.1;
%c = 14;

%% System Solver

%Differential Equation

%F = @(t,x) [-x(2)-x(3);x(1)+a*x(2);b+x(3)*(x(1)-c)];
%tspan = [0:delta_t:T];
%eps = 0.0000000001;
%options = odeset('RelTol',eps,'AbsTol',[eps eps eps/20]);
%[t,x] = ode45(F,tspan,x0,options);
%T = t(end);
%disp('Time actually ran for')
%disp(T)

%% Common Variables

%Calculating E (equation of interest phi)

E = 0.5 * (power(x(:,1),2) + power(x(:,2),2) + power(x(:,3),2));
% Derivatives

%dxdt = -x(:,2)-x(:,3);
%dydt = x(:,1)+a*x(:,2);
%dzdt = b+x(:,3).*(x(:,1)-c);

if p == 1; %Case A
    dxdt = -x(:,2);
    dydt = -x(:,1)+ x(:,2).*x(:,3);
    dzdt = 1 - x(:,2).^2;

elseif p == 2;   %Case B
        dxdt = x(:,2).*x(:,3);
        dydt = x(:,1)-x(:,2);
        dzdt = 1 - x(:,1).*x(:,2);

elseif p == 3;     %Case C
```

```
    dxdt = x(:,2).*x(:,3);
    dydt = x(:,1) - x(:,2);
    dzdt = 1 - x(:,1).^2 ;

elseif p == 4;  %Case F
    dxdt = x(:,2)+x(:,3);
    dydt = -x(:,1) +0.5*x(:,2);
    dzdt = x(:,1).^2-x(:,3);

elseif p == 5;  %Case E
    dxdt = x(:,2).*x(:,3);
    dydt = x(:,1).^2 -x(:,2);
    dzdt = 1-4*x(:,1);

end




%% Calculation of Average

%Step Average of E
Steps = round(T/delta_t);

%Integral of E
E_Sum = zeros([Steps,1]);
for i=1:Steps
    E_Sum(i+1) = E_Sum(i)+E(i+1)*delta_t;
end

E_bar = E_Sum(Steps)/T;

%% Calculation of Running Average
E_bar_run = zeros([Steps,1]);
for i = 1:Steps
    E_bar_run(i) = E_Sum(i)/(delta_t*i);
end

%% Calculating Standard Deviation
Std = zeros([Steps,1]);
for i =1:Steps
    Std(i) = (E(i)-E_bar);
end
Std_sum = sum(Std(i));

Std_bar = Std_sum/sqrt(T);

%% Calculating Running Standard Deviation
Std_run = zeros([Steps,1]);
for i = 1:Steps
    Std_run(i) = (E(i)-E_bar_run(i))/t(i);
end

%% Calculating Variance
Var = Std.^2;
Var_sum = Std_sum.^2;
Var_bar = Var_sum/T;

%% Calculating Running Variance
Var_run = Std_run.^2;
```

```
%% Specific Auxillary Lyapunov Function
Specific_Auxillary;

%% General Auxillary Lyapunov Function
General_Auxillary;

%% Sine Auxillary Function
Sine_Auxillary;

%% Minum point to Integrate to
%METHOD 1
%Linear method 1 - based on second order diffenrentials and integrating with respect to dt
%This method doesn't work as the stationary points are different for x y
%and z
%k1 = 0;
%k2 = 0;
%k3 = 0;
%x_min = [-1 1]*sqrt(c^2-k3);
%y_min = [-1 1].*sqrt(((x_min).^2 - k2)/a^2);
%z_min = [-1 1].*sqrt((y_min).^2 - k1);

%Second Method based on substituting first order differntials into the second order differntial

%for i = 1:Steps
%J(i) = b+x(i,3)*(x(i,1)-c)+(x(i,1)+a*x(i,2));
%K(i) = (-x(i,2)-x(i,3))+a*(x(i,1)+x(i,2));
%L(i) = x(i,1)*(b+x(i,3)*(x(i,1)-c))+x(i,3)*(-x(i,2)-x(i,3))-c*(b+x(i,3)*(x(i,1)-c));
%end

%J_min = mink(abs(J),2);
%K_min = mink(abs(K),2);
%L_min = mink(abs(L),2);
%Jmin_loc = find(J==-J_min);
%Kmin_loc = find(K==K_min);
%Lmin_loc = find(L==-L_min);

%Third method based on continous substitution to create an equation to find
%an approximate solution.
maxE = max(E);
minE = min(E);
maxE_loc = find(E == maxE);
minE_loc = find(E == minE);

for i =1:Steps
    if p == 1; %CaseA
        d2z(i) = dydt(i) - x(i,1) + x(i,2).*dzdt(i) + x(i,3).*dydt(i) - 2*x(i,3).*dydt(i);

    elseif p == 2; %Case B
        d2z(i) = x(i,2).*dzdt(i) + x(i,3).*dydt(i) + dxdt(i) - dydt(i) - x(i,1).*dydt(i) - x(i,2).*dxdt(i) ;

    elseif p == 3; %Case C
         d2z(i) = x(i,2).*dzdt(i) + x(i,3).*dydt(i) + dxdt(i) - dydt(i) - 2*x(i,1).*dxdt(i) ;

```

```matlab
    elseif p ==4; % Case F
        d2z(i) = dydt(i)+dzdt(i)-x(i,1)+0.5*x(i,2)+2*x(i,1).*dxdt(i)-dzdt(i);

    elseif p == 5; %Case E
        d2z(i) = x(i,2).*dzdt(i) + dzdt(i).*dydt(i) + 2*dxdt(i).*x(i,1)-dydt(i)
            - 4*dxdt(i);
    end

end
    %d2z(i) = a*(x(i,3)*dxdt(i))-x(i,1)*dzdt(i)-c*dzdt(i); for rossler

d2z_min = mink(abs(d2z),8);

d2zmin_loc(:,1) = find(abs(d2z) == d2z_min(1));
d2zmin_loc(:,2) = find(abs(d2z) == d2z_min(2));
d2zmin_loc(:,3) = find(abs(d2z) == d2z_min(3));
d2zmin_loc(:,4) = find(abs(d2z) == d2z_min(4));
d2zmin_loc(:,5) = find(abs(d2z) == d2z_min(5));
d2zmin_loc(:,6) = find(abs(d2z) == d2z_min(6));
%d2zmin_loc(:,7) = find(abs(d2z) == d2z_min(7));
%d2zmin_loc(:,8) = find(abs(d2z) == d2z_min(8));

min_d2z = find(d2zmin_loc < minE_loc);
if isempty(min_d2z) == 1
    min_d2z = minE_loc
    disp('minimum E value used for first stationairy point, not good')
end
min_d2z = max(min_d2z);
max_d2z = d2zmin_loc(d2zmin_loc > maxE_loc & d2zmin_loc> min_d2z);
max_d2z = min(max_d2z);

if isempty(max_d2z) == 1
    max_d2z = maxE_loc
    disp('maximum E value used for second stationary point, not good')
end


minT(:,1) = min_d2z;
minT(:,2) = max_d2z;



minT = sort(minT);
%Note that from this point onwards the value of Steps has Changed
Steps = minT(2)-minT(1); %new number of Steps
ind1 = minT(1);
ind2 = minT(2);
minT = Steps*delta_t; % reassigning Minimum T to the minimum time
t_m = t(ind1:ind2);
disp('The minimum time is')
disp(minT)

%% Calculation of Minimum Time Average

MinT_E = E(ind1:ind2);
%Integral of E
MinT_E_Sum = zeros([Steps,1]);
for i=1:Steps
    MinT_E_Sum(i+1) = MinT_E_Sum(i)+MinT_E(i)*delta_t;
end
```

```

MinT_E_bar = MinT_E_Sum(Steps)/(delta_t*Steps);

%% Calculation of Minimum Time Running Average
MinT_E_bar_run = zeros([Steps,1]);
for i = 1:Steps
    MinT_E_bar_run(i) = MinT_E_Sum(i)/(delta_t*i);
end

%% Calculating Variance - Specific
MinT_Var = zeros([Steps,1]);
for i =1:Steps
    MinT_Var(i) = (MinT_E(i)-MinT_E_bar)^2;
end
MinT_Var_sum = sum(MinT_Var(i));
MinT_Var_bar = MinT_Var_sum/(delta_t*Steps);

%% Calculating Running Variance - Specific
MinT_Var_run = zeros([Steps,1]);
for i = 1:Steps
    MinT_Var_run(i) = (MinT_E(i)-MinT_E_bar_run(i))^2;
end

%% Specific Auxillary Lyapunov Minimum Time
Specific_Auxillary_MinT;

%% General Auxillary Lypunov Minimum Time
General_Auxillary_minT;

%% Sine Auxillary Minimum Time
Sine_Auxillary_MinT;

%% Sum of Sigma = 0
min_std_run = min(abs(Std_run));
min_std_loc = find(abs(Std_run) == min_std_run);
E_std_run_0 = E_bar_run(min_std_loc);







%save('System_Base')
```

7.3.3 Specific Auxiliary

```
%Function to calculate the Specific Auxillary Case
%Created by Jamell Ivan Samuels

%% Finding the Specific Auxillary Function
%Calucltaing Lyapunov
%syms x1 x2 x3;
%xdot1 = -x2-x3;
%xdot2 = x1+a*x2;
%xdot3 = b+x3*(x1-c);
syms V(x1,x2, x3)
V(x1,x2,x3) = x1^2 + x1*x2 + x2^2 +x2*x3 +x1*x3 +x3^2
disp('Lypunov Generated')
%% Forming the Specific Lyapunov Auxillary Function
D1_spec = diff(V,x1); %(V_1=x)
```

```
D2_spec = diff(V,x2); %(V_2=y)
D3_spec = diff(V,x3);%(V_3=z)

x1 = x(:,1);
x2 = x(:,2);
x3 = x(:,3);
DD_spec(:,1) = subs(D1_spec);
DD_spec(:,2) = subs(D2_spec);
DD_spec(:,3) = subs(D3_spec);
DD_spec(:,1) = DD_spec(:,1).*dxdt;
DD_spec(:,2) = DD_spec(:,2).*dydt;
DD_spec(:,3) = DD_spec(:,3).*dzdt;
disp('DD matrices calculated')
%% Calculating Averages of DD
%% Average of DD
%Step Average of DD

%Integral of E + D
%Summation of D
DDSum_spec = zeros([Steps,1]);
for i = 1:Steps
DDSum_spec(i) = DD_spec(i,1)+DD_spec(i,2)+DD_spec(i,3);
end
DD_bar_spec = sum(DDSum_spec)/T;
%Calculating D^2 Bar
DD_spec_squared= zeros([Steps,1]);
for i = 1:Steps
    DD_spec_squared(i) = (DDSum_spec(i)).^2;
end
DD_spec_squared_Sum =  sum(DD_spec_squared);
DD_spec_squared_bar = DD_spec_squared_Sum/T; %DD Squared bar
%% Average of E & DD
%Combining of E & DD
Comb_E_D_spec = zeros([Steps,1]);
for i = 1:Steps
Comb_E_D_spec(i) = E(i)+DDSum_spec(i);
end

%Summation of E & DD
Sum_E_D_spec = zeros([Steps,1]);
for i = 1:Steps
    Sum_E_D_spec(i+1) = (Sum_E_D_spec(i))+(Comb_E_D_spec(i)*delta_t);
end

%Average of E + D

E_D_spec_bar = sum(Sum_E_D_spec(Steps))/T;

%% The Changing Average

for i = 1:Steps
E_D_spec_t(i) = sum(Sum_E_D_spec(i))/t(i); %Average of E + D specific
end


E_D_spec_t = transpose(E_D_spec_t);



%% Variance of Specific Auxillary
```

```
for i = 1:Steps
Var_spec(i) = (E_D_spec_t(i) - E_D_spec_bar).^2;
end

Var_spec_bar = sum(Var_spec(Steps))/T;


%% Calculation of alpha - Specific

for i = 1:Steps
    disp(i)
    alpha_spec(i) = (E(i)-E_bar)/(DD_spec_squared(i));
end

alpha_spec = alpha_spec/T;

alpha_spec = transpose(alpha_spec);

%alpha_test(1) =

%% Calculating Average of (E + alphaD) - Specific

%Combining E + alpha_bar D

E_D_spec_alpha = zeros([Steps,1]);
for i = 1:Steps
E_D_spec_alpha(i+1) = E_D_spec_alpha(i) + (E(i)+alpha_spec(i).*DDSum_spec(i))*
    delta_t;
end

%Averaging E +ialphaD


Sum_E_D_spec_alpha = sum(E_D_spec_alpha(Steps));
E_D_spec_alpha_bar = Sum_E_D_spec_alpha/T;

% Changing AVerage E +alpha D
for i = 1:Steps
    E_D_spec_alpha_t(i) = sum(E_D_spec_alpha(i))/t(i);
end

E_D_spec_alpha_t = transpose(E_D_spec_alpha_t);

%% Standard Deviation of E+alpha D

Std_alpha_spec = zeros([Steps,1]);
for i =1:Steps
    Std_alpha_spec(i) = (E_D_spec_alpha_t(i)-E_D_spec_alpha_bar);
end
Std_alpha_spec_sum = sum(Std_alpha_spec(Steps));
Std_alpha_spec_bar = Std_alpha_spec_sum/sqrt(T);


%% Variance of E+alpha  D

Var_alpha_spec = Std_alpha_spec.^2;
Var_alpha_spec_sum = Std_alpha_spec_sum.^2
Var_alpha_spec_bar = Var_alpha_spec_sum/T;
```

```
%% Calculating Running Alpha

for i = 1:Steps
    disp(i)
    alpha_spec_run(i) = (E(i)-E_bar_run(i))/(DD_spec_squared(i));
end

alpha_spec_run = alpha_spec_run/T;

alpha_spec_run = transpose(alpha_spec_run);

%% Calculating E +alpha D run

E_D_spec_alpha_run = zeros([Steps,1]);
for i = 1:Steps
E_D_spec_alpha_run(i+1) = E_D_spec_alpha_run(i)+(E(i)+alpha_spec_run(i).*(
    DDSum_spec(i)))*delta_t;
end

%Averaging E +ialphaD



Sum_E_D_spec_alpha_run = sum(E_D_spec_alpha_run(Steps));
E_D_spec_alpha_run_bar = Sum_E_D_spec_alpha_run/T;

for i = 1:Steps
    E_D_spec_alpha_run_t(i) = sum(E_D_spec_alpha_run(i))/t(i);
end

E_D_spec_alpha_run_t = transpose(E_D_spec_alpha_run_t);




%% Standard Deviation of E+alpha D run

Std_alpha_run_spec = zeros([Steps,1]);
for i =1:Steps
    Std_alpha_run_spec(i) = (E_D_spec_alpha_run_t(i)-E_D_spec_alpha_run_bar);
end
Std_alpha_run_spec_sum = sum(Std_alpha_run_spec(Steps));
Std_alpha_run_spec_bar = Std_alpha_run_spec_sum/sqrt(T);


%% Variance of E+alpha  D run
Var_alpha_run_spec = Std_alpha_run_spec.^2;
Var_alpha_run_spec_sum = Std_alpha_run_spec_sum.^2;
Var_alpha_run_spec_bar = Var_alpha_run_spec_sum/T;
```

7.3.4 General Auxiliary

```
%Function to calculate the Genera Auxillary Minimum Time Lyapunov Function

%Created by Jamell Ivan Samuels


%% Forming the General Auxillary Lyapunov Function


x1 = x(:,1);
```

```
x2 = x(:,2);
x3 = x(:,3);


DD_gen(:,1) = dxdt;
DD_gen(:,2) = dydt;
DD_gen(:,3) = dzdt;
DD_gen(:,4) = x(:,1).*dydt + x(:,2).*dxdt;
DD_gen(:,5) = x(:,1).*dzdt + x(:,3).*dxdt;
DD_gen(:,6) = x(:,3).*dydt + x(:,2).*dzdt;
DD_gen(:,7) = x(:,1).*x(:,3).*dydt + x(:,1).*x(:,2).*dzdt +x(:,2).*x(:,3).*dxdt;
DD_gen(:,8) = 2.*x(:,1).*dxdt;
DD_gen(:,9) = 2.*x(:,2).*dydt;
DD_gen(:,10) = 2.*x(:,3).*dzdt;
%% Calculating Averages of DD
%% Average of DD
%Step Average of DD

%Integral of E + D
%Summation of D
DDSum_gen = zeros([Steps,1]);
for i = 1:Steps
DDSum_gen(i) = DD_gen(i,1)+DD_gen(i,2)+DD_gen(i,3)+DD_gen(i,4)+DD_gen(i,5)+
    DD_gen(i,5)+DD_gen(i,6)+DD_gen(i,7)+DD_gen(i,8)+DD_gen(i,9)+DD_gen(i,10);
end
DD_bar_gen = sum(DDSum_gen)/T;
%Calculating D^2 Bar
DD_gen_squared= zeros([Steps,1]);
for i = 1:Steps
    DD_gen_squared(i) = (DDSum_gen(i)).^2;
end
DD_gen_squared_Sum =  sum(DD_gen_squared);
DD_gen_squared_bar = DD_gen_squared_Sum/T; %DD Squared bar
%% Average of E & DD
%Combining of E & DD
Comb_E_D_gen = zeros([Steps,1]);
for i = 1:Steps
Comb_E_D_gen(i) = E(i)+DDSum_gen(i);
end

%Summation of E & DD
Sum_E_D_gen = zeros([Steps,1]);
for i = 1:Steps
    Sum_E_D_gen(i+1) = (Sum_E_D_gen(i))+(Comb_E_D_gen(i)*delta_t);
end

%Average of E + D

for i = 1:Steps
    E_D_gen_t(i) = sum(Sum_E_D_gen(i))/t(i);
end
E_D_gen_t = transpose(E_D_gen_t);

E_D_gen_bar = sum(Sum_E_D_gen(Steps))/T; %Average of E + D specific

%% Variance of General Auxillary

for i = 1:Steps
    Var_gen(i) = ((E_D_gen_t(i)-E_D_gen_bar).^2)/(delta_t*i);
end
```

```
Var_gen_bar = sum(Var_gen(Steps))/T;


%% Calculation of alpha min - General
for i = 1:Steps
    disp(i)
    alpha_gen(i) = (E(i)-E_bar)/(DD_gen_squared(i));
end

alpha_gen = alpha_gen/T;

alpha_gen = transpose(alpha_gen);

%% Calculating Average of (E + alphaD) - General

%Combining E + alpha_bar D

E_D_gen_alpha = zeros([Steps,1]);
for i = 1:Steps
E_D_gen_alpha(i+1) = E_D_gen_alpha(i)+(E(i)+alpha_gen(i).*(DDSum_gen(i)))*delta_t;
end

%Averaging E +ialphaD
for i = 1:Steps
    E_D_gen_alpha_t(i) = sum(E_D_gen_alpha(i))/t(i);
end
E_D_gen_alpha_t = transpose(E_D_gen_alpha_t);


Sum_E_D_gen_alpha = sum(E_D_gen_alpha(Steps));
E_D_gen_alpha_bar = Sum_E_D_gen_alpha/T;



%% Standard Deviation of E+alpha  D

Std_alpha_gen = zeros([Steps,1]);
for i =1:Steps
    Std_alpha_gen(i) = (E_D_gen_alpha_t(i)-E_D_gen_alpha_bar)/(t(i));
end
Std_alpha_gen_sum = sum(Std_alpha_gen(Steps));
Std_alpha_gen_bar = Std_alpha_gen_sum/sqrt(T);

%% Variance of E +alpha D
Var_alpha_gen = Std_alpha_gen.^2;
Var_alpha_gen_sum = Std_alpha_gen_sum.^2;
Var_alpha_gen_bar = Var_alpha_gen_sum/T;

%% Calculating Running Alpha - General

for i = 1:Steps
    disp(i)
    alpha_gen_run(i) = (E(i)-E_bar_run(i))/(DD_gen_squared(i));
end

alpha_gen_run = alpha_gen_run/T;

alpha_gen_run = transpose(alpha_gen_run);
```

```
%% Calculating E +alpha D run - General

E_D_gen_alpha_run = zeros([Steps,1]);
for i = 1:Steps
E_D_gen_alpha_run(i+1) = E_D_gen_alpha_run(i) + (E(i)+alpha_gen_run(i).*(
    DDSum_gen(i)))*delta_t;
end

%Averaging E +ialphaD

for i = 1:Steps
    E_D_gen_alpha_run_t(i) = sum(E_D_gen_alpha_run(i))/t(i);
end


E_D_gen_alpha_run_t = transpose(E_D_gen_alpha_run_t);

Sum_E_D_gen_alpha_run = sum(E_D_gen_alpha_run(Steps));
E_D_gen_alpha_run_bar = Sum_E_D_gen_alpha_run/T;


%% Standard Deviation of E+alpha D run - General

Std_alpha_run_gen = zeros([Steps,1]);
for i =1:Steps
    Std_alpha_run_gen(i) = (E_D_gen_alpha_run_t(i)-E_D_gen_alpha_run_bar);
end
Std_alpha_run_gen_sum = sum(Std_alpha_run_gen(Steps));
Std_alpha_run_gen_bar = Std_alpha_run_gen_sum/sqrt(T);


%% Variance of E+alpha D run - General
Var_alpha_run_gen = Std_alpha_run_gen.^2;
Var_alpha_run_gen_sum = Std_alpha_run_gen_sum.^2;
Var_alpha_run_gen_bar = Var_alpha_run_gen_sum/T;


%%
```

7.3.5 Sine Auxiliary

```

%Function to calculate the Sine Auxillary Case
%Created by Jamell Ivan Samuels

%% Finding the Specific Auxillary Function

%% Forming the Sine Auxillary Function

E1 = E(1);
E2 = E(Steps);
for i = 1:Steps
    dEdt(i) = x(i,1)+x(i,2)+x(i,3)
end
for i = 1:Steps
DD_sine(i) = -dEdt(i).*cos((E1-E(i))/(E1-E2)*pi);
end
DD_sine = transpose(DD_sine)


```

```
%% Calculating Averages of DD
%% Average of DD
%Step Average of DD

%Integral of E + D
%Summation of D

DD_bar_sine = sum(DD_sine)/T;
%Calculating D^2 Bar
for i = 1:Steps
DD_sine_squared(i) = DD_sine(i).^2 ;
end
DD_sine_squared = transpose(DD_sine_squared);
DD_sine_squared_Sum =  sum(DD_sine_squared);
DD_sine_squared_bar = DD_sine_squared_Sum/T; %DD Squared bar
%% Average of E & DD
%Combining of E & DD
Comb_E_D_sine = zeros([Steps,1]);
for i = 1:Steps
Comb_E_D_sine(i) = E(i)+DD_sine(i);
end

%Summation of E & DD
Sum_E_D_sine = zeros([Steps,1]);
for i = 1:Steps
    Sum_E_D_sine(i+1) = Sum_E_D_sine(i)+Comb_E_D_sine(i)*delta_t;
end

%Average of E + D

for i = 1:Steps
    E_D_sine_t(i) = sum(Sum_E_D_sine(i))/t(i);
end

E_D_sine_t = transpose(E_D_sine_t);

E_D_sine_bar = sum(Sum_E_D_sine(Steps))/T; %Average of E + D specific

%% Variance of Sine Auxillary

for i = 1:Steps
Var_sine(i) = (E_D_sine_t(i) - E_D_sine_bar).^2;
end

Var_sine_bar = sum(Var_sine(Steps))/T;

%% Calculation of alpha min - Sine

for i = 1:Steps
    disp(i)
    alpha_sine(i) = (E(i)-E_bar)/(DD_sine_squared(i));
end

alpha_sine = alpha_sine/T;

alpha_sine = transpose(alpha_sine);
%alpha_sine(1) = 0; % THis actually doesn't make
%alpha_sine(end) = 0;
```

```
%% Calculating Average of (E + alphaD) - Sine

%Combining E + alpha_bar D

E_D_sine_alpha = zeros([Steps,1]);
for i = 1:Steps
E_D_sine_alpha(i+1) = E_D_sine_alpha(i)+(E(i)+alpha_sine(i).*DD_sine(i))*delta_t;
end

%Averaging E +ialphaD

for i = 1:Steps
    E_D_sine_alpha_t(i) = sum(E_D_sine_alpha(i))/t(i);
end
E_D_sine_alpha_t = transpose(E_D_sine_alpha_t);

Sum_E_D_sine_alpha = sum(E_D_sine_alpha(Steps));
E_D_sine_alpha_bar = Sum_E_D_sine_alpha/T;

%% Standard Deviation of E +alpha D

Std_alpha_sine = zeros([Steps,1]);
for i =1:Steps
    Std_alpha_sine(i) = (E_D_sine_alpha_t(i)-E_D_sine_alpha_bar); % This is is the sample deviation
end
Std_alpha_sine_sum = sum(Std_alpha_sine(Steps));
Std_alpha_min_sine_bar = Std_alpha_sine_sum/sqrt(T);  %This is the standard deviation


%% Variance of E+alpha D
Var_alpha_sine = Std_alpha_sine.^2;
Var_alpha_sine_sum = Std_alpha_sine_sum.^2;
Var_alpha_sine_bar = Var_alpha_sine_sum/T;

%% Calculating Running Alpha

for i = 1:Steps
    disp(i)
    alpha_sine_run(i) = (E(i)-E_bar_run(i))/(DD_sine_squared(i));
end

alpha_sine_run = alpha_sine_run/T;

alpha_sine_run = transpose(alpha_sine_run);

alpha_sine_run(1) = 0;

alpha_sine_run(end) = 0;
%% Calculating E +alpha D run

E_D_sine_alpha_run = zeros([Steps,1]);
for i = 1:Steps
E_D_sine_alpha_run(i+1) = E_D_sine_alpha_run(i)+(E(i)+alpha_sine_run(i).*DD_sine(i))*delta_t;
end

%Averaging E +ialphaD
```

```
for i = 1:Steps
    E_D_sine_alpha_run_t(i) = sum(E_D_sine_alpha_run(i))/t(i);
end

E_D_sine_alpha_run_t = transpose(E_D_sine_alpha_run_t);

Sum_E_D_sine_alpha_run = sum(E_D_sine_alpha_run(Steps));
E_D_sine_alpha_run_bar = Sum_E_D_sine_alpha_run/T;


%% Standard Deviation of E+alpha D run

Std_alpha_run_sine = zeros([Steps,1]);
for i =1:Steps
    Std_alpha_run_sine(i) = (E_D_sine_alpha_run_t(i)-E_D_sine_alpha_run_bar);
end
Std_alpha_run_sine_sum = sum(Std_alpha_run_sine(Steps));
Std_alpha_run_sine_bar = Std_alpha_run_sine_sum/sqrt(T);


%% Variance of E+alpha  D run
Var_alpha_run_sine = Std_alpha_run_sine.^2;
Var_alpha_run_sine_sum = Std_alpha_run_sine_sum.^2;
Var_alpha_run_sine_bar = Var_alpha_run_sine_sum/T;

%% Changing Minimum Time Average
```

7.3.6 Minimum Time Specific Auxiliary

```
%Function to calculate the Specific Auxillary Miniumum Time Case
%Created by Jamell Ivan Samuels

%% Finding the Specific Auxillary Function
%Calucltaing Lyapunov

%% Forming the Specific Lyapunov Auxillary Function

x1 = x(ind1:ind2,1);
x2 = x(ind1:ind2,2);
x3 = x(ind1:ind2,3);


MinT_DD_spec(:,1) = DD_spec(ind1:ind2,1);
MinT_DD_spec(:,2) = DD_spec(ind1:ind2,2);
MinT_DD_spec(:,3) = DD_spec(ind1:ind2,3);

%% Calculating Averages of DD
%% Average of DD
%Step Average of DD

%Integral of E + D
%Summation of D
MinT_DDSum_spec = zeros([Steps,1]);
for i = 1:Steps
MinT_DDSum_spec(i) = MinT_DD_spec(i,1)+MinT_DD_spec(i,2)+MinT_DD_spec(i,3);
end
MinT_DD_bar_spec = sum(MinT_DDSum_spec)/minT;
%Calculating D^2 Bar
MinT_DD_spec_squared= zeros([Steps,1]);
for i = 1:Steps
```

```
    MinT_DD_spec_squared(i) = MinT_DDSum_spec(i).^2;
end
MinT_DD_spec_squared_Sum =  sum(MinT_DD_spec_squared);
MinT_DD_spec_squared_bar = MinT_DD_spec_squared_Sum/minT; %DD Squared bar
%% Average of E & DD
%Combining of E & DD
MinT_Comb_E_D_spec = zeros([Steps,1]);
for i = 1:Steps
MinT_Comb_E_D_spec(i) = MinT_E(i)+MinT_DDSum_spec(i);
end

%Summation of E & DD
MinT_Sum_E_D_spec = zeros([Steps,1]);
for i = 1:Steps
    MinT_Sum_E_D_spec(i+1) = (MinT_Sum_E_D_spec(i))+(MinT_Comb_E_D_spec(i)*
        delta_t);
end

%Average of E + D
for i = 1:Steps
    MinT_E_D_spec_t(i) = sum(MinT_Sum_E_D_spec(i))/t_m(i);
end

MinT_E_D_spec_t = transpose(MinT_E_D_spec_t);
MinT_E_D_spec_bar = sum(MinT_Sum_E_D_spec(Steps))/minT; %Average of E + D
    specific


%% Variance of Specific Auxillary Minimum Time

for i = 1:Steps
MinT_Var_spec(i) = (MinT_E_D_spec_t(i) - MinT_E_D_spec_bar).^2;
end

MinT_Var_spec_bar = sum(MinT_Var_spec(Steps))/minT;

%% Calculation of alpha - General

for i = 1:Steps
    disp(i)
    MinT_alpha_spec(i) = (MinT_E(i)-MinT_E_bar)/(MinT_DD_spec_squared(i));
end

MinT_alpha_spec = MinT_alpha_spec/minT;

MinT_alpha_spec = transpose(MinT_alpha_spec);

%% Calculating Average of (E + alphaD) - Specific

%Combining E + alpha_bar D

MinT_E_D_spec_alpha = zeros([Steps,1]);
for i = 1:Steps
MinT_E_D_spec_alpha(i+1) = MinT_E_D_spec_alpha(i)+(MinT_E(i)+MinT_alpha_spec(i)
    .*MinT_DDSum_spec(i))*delta_t;
end

%Averaging E +ialphaD

for i = 1:Steps
```

```
    MinT_E_D_spec_alpha_t(i) = sum(MinT_E_D_spec_alpha(i))/t_m(i);
end
MinT_E_D_spec_alpha_t = transpose(MinT_E_D_spec_alpha_t);

MinT_Sum_E_D_spec_alpha = sum(MinT_E_D_spec_alpha(Steps));
MinT_E_D_spec_alpha_bar = MinT_Sum_E_D_spec_alpha/minT;


%% Standard Deviation of E+alpha_min  D

MinT_Std_alpha_spec = zeros([Steps,1]);
for i =1:Steps
    MinT_Std_alpha_spec(i) = (MinT_E_D_spec_alpha_t(i)-MinT_E_D_spec_alpha_bar);
end
MinT_Std_alpha_spec_sum = sum(MinT_Std_alpha_spec(Steps));
MinT_Std_alpha_spec_bar = MinT_Std_alpha_spec_sum/minT;

%% Variance of E +alpha min D
MinT_Var_alpha_spec = MinT_Std_alpha_spec.^2;
MinT_Var_alpha_spec_sum = MinT_Std_alpha_spec_sum.^2;
MinT_Var_alpha_spec_bar = MinT_Var_alpha_spec_sum/minT;

%% Calculating Running Alpha

for i = 1:Steps
    disp(i)
    MinT_alpha_spec_run(i) = (MinT_E(i)-MinT_E_bar_run(i))/(MinT_DD_spec_squared
        (i));
end

MinT_alpha_spec_run = MinT_alpha_spec_run/minT;

MinT_alpha_spec_run = transpose(MinT_alpha_spec_run);

%% Calculating E +alpha D run

MinT_E_D_spec_alpha_run = zeros([Steps,1]);
for i = 1:Steps
MinT_E_D_spec_alpha_run(i+1) = MinT_E_D_spec_alpha_run(i)+(MinT_E(i)+
    MinT_alpha_spec_run(i).*(MinT_DDSum_spec(i)))*delta_t;
end

%Averaging E +ialphaD

for i = 1:Steps
    MinT_E_D_spec_alpha_run_t(i) = sum(MinT_E_D_spec_alpha_run(i))/t_m(i);
end

MinT_E_D_spec_alpha_run_t = transpose(MinT_E_D_spec_alpha_run_t);

MinT_Sum_E_D_spec_alpha_run = sum(MinT_E_D_spec_alpha_run(Steps));
MinT_E_D_spec_alpha_run_bar = MinT_Sum_E_D_spec_alpha_run/minT;


%% Standard Deviation of E+alpha D run

MinT_Std_alpha_run_spec = zeros([Steps,1]);
for i =1:Steps
    MinT_Std_alpha_run_spec(i) = (MinT_E_D_spec_alpha_run_t(i)-
        MinT_E_D_spec_alpha_run_bar);
```

```
end
MinT_Std_alpha_run_spec_sum = sum(MinT_Std_alpha_run_spec(Steps));
MinT_Std_alpha_run_spec_bar = MinT_Std_alpha_run_spec_sum/sqrt(minT);


%% Variance of E+alpha D run
MinT_Var_alpha_run_spec = MinT_Std_alpha_run_spec.^2;
MinT_Var_alpha_run_spec_sum = MinT_Std_alpha_run_spec_sum.^2;
MinT_Var_alpha_run_spec_bar = MinT_Var_alpha_run_spec_sum/minT;

%% Changing Minimum Time Average
```

7.3.7 Minimum Time General Auxiliary

```
%Function to calculate the Genera AUxillary Lyapunov Function

%Created by Jamell Ivan Samuels


%% Forming the General Auxillary Lyapunov Function


x1 = x(:,1);
x2 = x(:,2);
x3 = x(:,3);


MinT_DD_gen(:,1) = dxdt(ind1:ind2);
MinT_DD_gen(:,2) = dydt(ind1:ind2);
MinT_DD_gen(:,3) = dzdt(ind1:ind2);
MinT_DD_gen(:,4) = x(ind1:ind2,1).*dydt(ind1:ind2) + x(ind1:ind2,2).*dxdt(ind1:
    ind2);
MinT_DD_gen(:,5) = x(ind1:ind2,1).*dzdt(ind1:ind2,1) + x(ind1:ind2,3).*dxdt(ind1
    :ind2);
MinT_DD_gen(:,6) = x(ind1:ind2,3).*dydt(ind1:ind2) + x(ind1:ind2,2).*dzdt(ind1:
    ind2);
MinT_DD_gen(:,7) = x(ind1:ind2,1).*x(ind1:ind2,3).*dydt(ind1:ind2) + x(ind1:ind2
    ,1).*x(ind1:ind2,2).*dzdt(ind1:ind2) +x(ind1:ind2,2).*x(ind1:ind2,3).*dxdt(
    ind1:ind2);
MinT_DD_gen(:,8) = 2.*x(ind1:ind2,1).*dxdt(ind1:ind2);
MinT_DD_gen(:,9) = 2.*x(ind1:ind2,2).*dydt(ind1:ind2);
MinT_DD_gen(:,10) = 2.*x(ind1:ind2,3).*dzdt(ind1:ind2);



%% Calculating Averages of DD
%% Average of DD
%Step Average of DD

%Integral of E + D
%Summation of D
MinT_DDSum_gen = zeros([Steps,1]);
for i = 1:Steps
MinT_DDSum_gen(i) = MinT_DD_gen(i,1)+MinT_DD_gen(i,2)+MinT_DD_gen(i,3)+
    MinT_DD_gen(i,4)+MinT_DD_gen(i,5)+MinT_DD_gen(i,5)+MinT_DD_gen(i,6)+
    MinT_DD_gen(i,7)+MinT_DD_gen(i,8)+MinT_DD_gen(i,9)+MinT_DD_gen(i,10);
end
MinT_DD_bar_gen = sum(MinT_DDSum_gen)/minT;
%Calculating D^2 Bar
MinT_DD_gen_squared= zeros([Steps,1]);
for i = 1:Steps
```

```
    MinT_DD_gen_squared(i) = (MinT_DDSum_gen(i)).^2;
end
MinT_DD_gen_squared_Sum =  sum(MinT_DD_gen_squared);
MinT_DD_gen_squared_bar = MinT_DD_gen_squared_Sum/minT; %DD Squared bar
%% Average of E & DD
%Combining of E & DD
MinT_Comb_E_D_gen = zeros([Steps,1]);
for i = 1:Steps
MinT_Comb_E_D_gen(i) = E(i)+MinT_DDSum_gen(i);
end

%Summation of E & DD
MinT_Sum_E_D_gen = zeros([Steps,1]);
for i = 1:Steps
    MinT_Sum_E_D_gen(i+1) = (MinT_Sum_E_D_gen(i))+(MinT_Comb_E_D_gen(i)*delta_t)
        ;
end

%Average of E + D
for i = 1:Steps
    MinT_E_D_gen_t(i) = sum(MinT_Sum_E_D_gen(i))/t_m(i);
end
MinT_E_D_gen_t = transpose(MinT_E_D_gen_t);


MinT_E_D_gen_bar = sum(MinT_Sum_E_D_gen(Steps))/minT; %Average of E + D specific


%% Variance of General Auxillary Minimum Time

for i = 1:Steps
MinT_Var_gen(i) = (MinT_E_D_gen_t(i) - MinT_E_D_gen_bar).^2;
end

MinT_Var_gen_bar = sum(MinT_Var_gen(Steps))/minT;




%% Calculation of alpha - Genaral

for i = 1:Steps
    disp(i)
    MinT_alpha_gen(i) = (MinT_E(i)-MinT_E_bar)/(MinT_DD_gen_squared(i));
end

MinT_alpha_gen = MinT_alpha_gen/minT;

MinT_alpha_gen = transpose(MinT_alpha_gen);

%% Calculating Average of (E + alpha_minD) - Specific

%Combining E + alpha D

MinT_E_D_gen_alpha = zeros([Steps,1]); %combs are e+alpha d
for i = 1:Steps
MinT_E_D_gen_alpha(i+1) = MinT_E_D_gen_alpha(i)+(MinT_E(i)+MinT_alpha_gen(i).*
    MinT_DDSum_gen(i))*delta_t;
end

```

```
%Averaging E +ialphaD

for i = 1:Steps
    MinT_E_D_gen_alpha_t(i) = sum(MinT_E_D_gen_alpha(i))/t_m(i);
end
MinT_E_D_gen_alpha_t = transpose(MinT_E_D_gen_alpha_t);

MinT_Sum_E_D_gen_alpha = sum(MinT_E_D_gen_alpha(Steps));
MinT_E_D_gen_alpha_bar = MinT_Sum_E_D_gen_alpha/minT;

%% Calculating Runing Alpha - Specific

for i = 1:Steps
    disp(i)
    MinT_alpha_gen_run(i) = (E(i)-E_bar_run(i))/(DD_gen_squared(i));
end

MinT_alpha_gen_run = MinT_alpha_gen/minT;

%MinT_alpha_gen = transpose(MinT_alpha_gen)


%% Standard Deviation of E+alpha_min  D

MinT_Std_alpha_gen = zeros([Steps,1]);
for i =1:Steps
    MinT_Std_alpha_gen(i) = (MinT_E_D_gen_alpha_t(i)-MinT_E_D_gen_alpha_bar);
end
MinT_Std_alpha_gen_sum = sum(MinT_Std_alpha_gen(Steps));
MinT_Std_alpha_gen_bar = MinT_Std_alpha_gen_sum/sqrt(minT);


%% Variance of E +alpha min  D
MinT_Var_alpha_gen = MinT_Std_alpha_gen.^2;
MinT_Var_alpha_gen_sum = MinT_Std_alpha_gen_sum.^2;
MinT_Var_alpha_gen_bar = MinT_Var_alpha_gen_sum/minT;

%% Calculating Running Alpha

for i = 1:Steps
    disp(i)
    MinT_alpha_gen_run(i) = (MinT_E(i)-MinT_E_bar_run(i))/(MinT_DD_gen_squared(i
        ));
end

MinT_alpha_gen_run = MinT_alpha_gen_run/minT;

%MinT_alpha_gen_run = transpose(MinT_alpha_gen_run)

%% Calculating E +alpha D run

MinT_E_D_gen_alpha_run = zeros([Steps,1]);
for i = 1:Steps
MinT_E_D_gen_alpha_run(i+1) = MinT_E_D_gen_alpha_run(i)+(MinT_E(i)+
    MinT_alpha_gen_run(i).*(MinT_DDSum_gen(i)))*delta_t;
end

%Averaging E +ialphaD


```

```
for i = 1:Steps
    MinT_E_D_gen_alpha_run_t(i) = sum(MinT_E_D_gen_alpha_run(i))/t_m(i);
end


MinT_E_D_gen_alpha_run_t = transpose(MinT_E_D_gen_alpha_run_t);

MinT_Sum_E_D_gen_alpha_run = sum(MinT_E_D_gen_alpha_run(Steps));
MinT_E_D_gen_alpha_run_bar = MinT_Sum_E_D_gen_alpha_run/minT;


%% Standard Deviation of E+alpha D run

MinT_Std_alpha_run_gen = zeros([Steps,1]);
for i =1:Steps
    MinT_Std_alpha_run_gen(i) = (MinT_E_D_gen_alpha_run_t(i)-
        MinT_E_D_gen_alpha_run_bar);
end
MinT_Std_alpha_run_gen_sum = sum(MinT_Std_alpha_run_gen(Steps));
MinT_Std_alpha_run_gen_bar = MinT_Std_alpha_run_gen_sum/sqrt(minT);


%% Variance of E+alpha  D run
MinT_Var_alpha_run_gen = MinT_Std_alpha_run_gen.^2;
MinT_Var_alpha_run_gen_sum = MinT_Std_alpha_run_gen_sum.^2;
MinT_Var_alpha_run_gen_bar = MinT_Var_alpha_run_gen_sum/minT;


%% Changing Minimum Time Average
```

7.3.8 Minimum Time Sine Auxiliary

```
%Function to calculate the Minimum Time Sine Auxillary Case
%Created by Jamell Ivan Samuels

%% Finding the Specific Auxillary Function

%% Forming the Sine Auxillary Function


MinT_DD_sine = DD_sine(ind1:ind2);


%% Calculating Averages of DD
%% Average of DD
%Step Average of DD

%Integral of E + D
%Summation of D
MinT_DDSum_sine = sum(MinT_DD_sine);

MinT_DD_bar_sine = sum(MinT_DDSum_sine)/minT;
%Calculating D^2 Bar
for i = 1:Steps
MinT_DD_sine_squared(i) = (MinT_DD_sine(i)).^2;
end
MinT_DD_sine_squared_Sum =  sum(MinT_DD_sine_squared);
MinT_DD_sine_squared_bar = MinT_DD_sine_squared_Sum/minT; %DD Squared bar
%% Average of E & DD
%Combining of E & DD
MinT_Comb_E_D_sine = zeros([Steps,1]);
```

```
for i = 1:Steps
MinT_Comb_E_D_sine(i) = MinT_E(i)+MinT_DD_sine(i);
end

%Summation of E & DD
MinT_Sum_E_D_sine = zeros([Steps,1]);
for i = 1:Steps
    MinT_Sum_E_D_sine(i+1) = (MinT_Sum_E_D_sine(i))+(MinT_Comb_E_D_sine(i)*
        delta_t);
end

%Average of E + D

for i = 1:Steps
    MinT_E_D_sine_t(i) = sum(MinT_Sum_E_D_sine(i))/t_m(i);
end
MinT_E_D_sine_t = transpose(MinT_E_D_sine_t);


MinT_E_D_sine_bar = sum(MinT_Sum_E_D_sine(Steps))/minT; %Average of E + D
    specific


%% Variance of Sine Auxillary Minimum Time

for i = 1:Steps
MinT_Var_sine(i) = (MinT_E_D_sine_t(i) - MinT_E_D_sine_bar).^2;
end

MinT_Var_sine_bar = sum(MinT_Var_sine(Steps))/minT;



%% Calculation of alpha min - Sine

for i = 1:Steps
    disp(i)
    MinT_alpha_sine(i) = (MinT_E(i)-MinT_E_bar)/(MinT_DD_sine_squared(i));
end

MinT_alpha_sine = MinT_alpha_sine/minT;

MinT_alpha_sine = transpose(MinT_alpha_sine);


%% Calculating Average of (E + alphaD) - Sine

%Combining E + alpha_bar D

MinT_E_D_sine_alpha = zeros([Steps,1]);
for i = 1:Steps
MinT_E_D_sine_alpha(i+1) = MinT_E_D_sine_alpha(i) + (MinT_E(i) + MinT_alpha_sine
    (i).*MinT_DD_sine(i))*delta_t;
end

%Averaging E +ialphaD

for i = 1:Steps
    MinT_E_D_sine_alpha_t(i) = sum(MinT_E_D_sine_alpha(i))/t_m(i);
end
```

```
MinT_E_D_sine_alpha_t = transpose(MinT_E_D_sine_alpha_t);

MinT_Sum_E_D_sine_alpha = sum(MinT_E_D_sine_alpha(Steps));
MinT_E_D_sine_alpha_bar = MinT_Sum_E_D_sine_alpha/minT;

%% Standard Deviation of E +alpha D

MinT_Std_alpha_sine = zeros([Steps,1]);
for i =1:Steps
    MinT_Std_alpha_sine(i) = (MinT_E_D_sine_alpha_t(i)-MinT_E_D_sine_alpha_bar); % This is is the sample deviation
end
MinT_Std_alpha_sine_sum = sum(MinT_Std_alpha_sine(Steps));
MinT_Std_alpha_min_sine_bar = MinT_Std_alpha_sine_sum/sqrt(minT);  %This is the standard deviation


%% Variance of E+alpha D
MinT_Var_alpha_sine = MinT_Std_alpha_sine.^2;
MinT_Var_alpha_sine_sum = MinT_Std_alpha_sine_sum.^2;
MinT_Var_alpha_sine_bar = MinT_Var_alpha_sine_sum/minT;

%% Calculating Running Alpha

for i = 1:Steps
    disp(i)
    MinT_alpha_sine_run(i) = (MinT_E(i)-MinT_E_bar_run(i))/(MinT_DD_sine_squared(i));
end

MinT_alpha_sine_run = MinT_alpha_sine_run/minT;

MinT_alpha_sine_run = transpose(MinT_alpha_sine_run);

%% Calculating E +alpha D run

MinT_E_D_sine_alpha_run = zeros([Steps,1]);
for i = 1:Steps
MinT_E_D_sine_alpha_run(i+1) = MinT_E_D_sine_alpha_run(i) +(MinT_E(i)+MinT_alpha_sine_run(i).*(MinT_DD_sine(i)))*delta_t;
end

%Averaging E +ialphaD

for i = 1:Steps
    MinT_E_D_sine_alpha_run_t(i) = sum(MinT_E_D_sine_alpha_run(i))/t_m(i);
end

MinT_E_D_sine_alpha_run_t = transpose(MinT_E_D_sine_alpha_run_t);

MinT_Sum_E_D_sine_alpha_run = sum(MinT_E_D_sine_alpha_run(Steps));
MinT_E_D_sine_alpha_run_bar = MinT_Sum_E_D_sine_alpha_run/minT;


%% Standard Deviation of E+alpha D run

MinT_Std_alpha_run_sine = zeros([Steps,1]);
for i =1:Steps
    MinT_Std_alpha_run_sine(i) = (MinT_E_D_sine_alpha_run_t(i)-MinT_E_D_sine_alpha_run_bar);
```

```
end
MinT_Std_alpha_run_sine_sum = sum(MinT_Std_alpha_run_sine(Steps));
MinT_Std_alpha_run_sine_bar = MinT_Std_alpha_run_sine_sum/sqrt(minT);


%% Variance of E+alpha  D run
MinT_Var_alpha_run_sine = MinT_Std_alpha_run_sine.^2;
MinT_Var_alpha_run_sine_sum = MinT_Std_alpha_run_sine_sum.^2;
MinT_Var_alpha_run_sine_bar = MinT_Var_alpha_run_sine_sum/minT;

%% Changing Minimum Time Average
```

7.3.9 Plot Function

```
% Function created to plot diagrams
%Created by Jamell Ivan Samuels

%% Plotting All Averages
%Specific
plot(t,ones([10001,1])*4.1803,t,[0;E_bar_run],t,[0;E_D_spec_t],'--',t,[0;E_D_spec_alpha_t],':',t,[0;E_D_spec_alpha_run_t],'-.');
xlabel('time (s)')
u = ylabel('Quantity of interest ($$\overline{\Phi}$$)')
set(u,'Interpreter','latex','fontsize',12)
h = legend('$$\overline{E_{true}}$$','$$\overline{E}$$','$$\overline{E+D}$$','$$\overline{E+\alpha_{min}D}$$','$$\overline{E+\alpha_{run}D}$$','Location','Best')
set(h,'Interpreter','latex','fontsize',12)
print(gcf,'Specific_Averages.png','-dpng')
%General
plot(t,ones([10001,1])*4.1803,t,[0;E_bar_run],t,[0;E_D_gen_t],'--',t,[0;E_D_gen_alpha_t],':',t,[0;E_D_gen_alpha_run_t],'-.');
xlabel('time (s)')
u = ylabel('Quantity of interest ($$\overline{\Phi}$$)')
set(u,'Interpreter','latex','fontsize',12)
h = legend('$$\overline{E_{true}}$$','$$\overline{E+D}$$','$$\overline{E+\alpha_{min}D}$$','$$\overline{E+\alpha_{run}D}$$','Location','Best')
set(h,'Interpreter','latex','fontsize',12)
print(gcf,'General_Averages.png','-dpng')
%Sine
plot(t,ones([10001,1])*4.1803,t,[0;E_bar_run],t,[0;E_D_sine_t],'--',t,[0;E_D_sine_alpha_t],':',t,[0;E_D_sine_alpha_run_t],'-.');
xlabel('time (s)')
u = ylabel('Quantity of interest ($$\overline{\Phi}$$)')
set(u,'Interpreter','latex','fontsize',12)
h = legend('$$\overline{E_{true}}$$','$$\overline{E}$$','$$\overline{E+D}$$','$$\overline{E+\alpha_{min}D}$$','$$\overline{E+\alpha_{run}D}$$','Location','Best')
set(h,'Interpreter','latex','fontsize',12)
print(gcf,'Sine_Averages.png','-dpng')

%% Plot Side by Side Average Comparison
%Base Systems
plot(t,ones([10001,1])*4.1803,t,[0;E_bar_run],t,[0;E_D_spec_t],'--',t,[0;E_D_gen_t],':',t,[0;E_D_sine_t],'-.');
xlabel('time (s)')
u = ylabel('Quantity of interest ($$\overline{\Phi}$$)')
set(u,'Interpreter','latex','fontsize',12)
legend('True','Base','Specific','General','Sine','Location','Best')
print(gcf,'Base_Averages.png','-dpng')

```

```
%alpha Systems
plot(t,ones([10001,1])*4.1803,t,[0;E_D_spec_alpha_t],'--',t,[0;E_D_gen_alpha_t],
    ':',t,[0;E_D_sine_alpha_t],'-.');
xlabel('time (s)')
u = ylabel('Quantity of interest ($$\overline{\Phi}$$)')
set(u,'Interpreter','latex','fontsize',12)
legend('True','Specific','General','Sine','Location','Best')
print(gcf,'Alpha_Averages.png','-dpng')

%alpha run Systems

plot(t,ones([10001,1])*4.1803,t,[0;E_D_spec_alpha_run_t],'--',t,[0;
    E_D_gen_alpha_run_t],':',t,[0;E_D_sine_alpha_run_t],'-.');
xlabel('time (s)')
u = ylabel('Quantity of interest ($$\overline{\Phi}$$)')
legend('True','Specific','General','Sine','Location','Best')
set(u,'Interpreter','latex','fontsize',12)
print(gcf,'Alpha_Run_Averages.png','-dpng')

%% Plot All Variances

%Specific
plot(t,[0;Var],':',t,[0,Var_spec],'--',t,[0;Var_alpha_spec],'-.',t,[0;
    Var_alpha_run_spec]);
xlabel('time (s)')
ylabel('Variance (\sigma^2)')
legend('\sigma^2','\sigma_{spec}^2','\sigma_{spec alpha}^2','\sigma_{spec alpha
    run}^2','Location','Best')
print(gcf,'Specific_Variances.png','-dpng')
%General
plot(t,[0;Var],':',t,[0,Var_gen],'--',t,[0;Var_alpha_gen],'-.',t,[0;
    Var_alpha_run_gen]);
xlabel('time (s)')
ylabel('Variance (\sigma^2)')
legend('\sigma^2','\sigma_{gen}^2','\sigma_{gen alpha}^2','\sigma_{gen alpha run
    }^2','Location','Best')
print(gcf,'General_Variances.png','-dpng')
%Sine
plot(t,[0;Var],':',t,[0,Var_sine],'--',t,[0;Var_alpha_sine],'-.',t,[0;
    Var_alpha_run_sine]);
xlabel('time (s)')
ylabel('Variance (\sigma^2)')
legend('\sigma^2','\sigma_{sine}^2','\sigma_{sine alpha}^2','\sigma_{sine alpha
    run}^2','Location','Best')
print(gcf,'Sine_Variances.png','-dpng')



%% Plot Side by Side Variance Comparison
%Base
plot(t,[0;Var],'--',t,[0,Var_spec],':',t,[0,Var_gen],'-.',t,[0,Var_sine]);
xlabel('time (s)')
ylabel('Variance (\sigma^2)')
legend('\sigma^2','\sigma_{spec}^2','\sigma_{gen}^2','\sigma_{sine}^2','Location
    ','Best')
print(gcf,'Base_Variances.png','-dpng')

%alpha
plot(t,[0;Var],'--',t,[0;Var_alpha_spec],':',t,[0;Var_alpha_gen],'-.',t,[0;
    Var_alpha_sine]);
```

```
xlabel('time (s)')
ylabel('Variance (\sigma^2)')
legend('\sigma^2','\sigma_{spec}^2','\sigma_{gen}^2','\sigma_{sine}^2','Location','Best')
print(gcf,'Alpha_Variances.png','-dpng')

%alpha run
plot(t,[0;Var],'--',t,[0;Var_alpha_run_spec],':',t,[0;Var_alpha_run_gen],'-.',t,[0;Var_alpha_run_sine]);
xlabel('time (s)')
ylabel('Variance \sigma^2')
legend('\sigma^2','\sigma_{spec}^2','\sigma_{gen}^2','\sigma_{sine}^2','Location','Best')
print(gcf,'Alpha_Run_Variances_.png','-dpng')

%% Plot Side by Side alpha vs alpha run comparison
%all alphas

plot(t,[0;alpha_spec],'--',t,[0;alpha_gen],':',t,[0;alpha_sine],'-.');
xlabel('time (s)')
ylabel('\alpha')
legend('\alpha_{spec}','\alpha_{gen}','\alpha_{sine}','Location','Best')
print(gcf,'Alpha.png','-dpng')


%all alpha runs
plot(t,[0;alpha_spec_run],'--',t,[0;alpha_gen_run],':',t,[0;alpha_sine_run],'-.');
xlabel('time (s)')
ylabel('\alpha')
legend('\alpha_{spec}','\alpha_{gen}','\alpha_{sine}','Location','Best')
print(gcf,'Alpha_Run.png','-dpng')

%specs

plot(t,[0;alpha_spec],'--',t,[0;alpha_spec_run],':');
xlabel('time (s)')
ylabel('\alpha')
legend('\alpha_{spec}','\alpha_{spec run}','Location','Best')
print(gcf,'Alpha_Spec_.png','-dpng')

%gens
plot(t,[0;alpha_gen],'--',t,[0;alpha_gen_run],':');
xlabel('time (s)')
ylabel('\alpha')
legend('\alpha_{gen}','\alpha_{gen run}','Location','Best')
print(gcf,'Alpha_Gen_.png','-dpng')

%sines

plot(t,[0;alpha_sine],'--',t,[0;alpha_sine_run],':');
xlabel('time (s)')
ylabel('\alpha')
legend('\alpha_{sine}','\alpha_{sine run}','Location','Best')
print(gcf,'Alpha_Sine_.png','-dpng')
```

7.3.10 Minimum Time Plot Function

```
% Function created to plot diagrams
%Created by Jamell Ivan Samuels

```

```
%% Plotting All Averages
%Specific
plot(t_m,ones([Steps+1,1])*4.1803,t_m,[0;MinT_E_bar_run],t_m,[0;MinT_E_D_spec_t],'--',t_m,[0;MinT_E_D_spec_alpha_t],':',t_m,[0;MinT_E_D_spec_alpha_run_t],'-.');
xlabel('time (s)')
u = ylabel('Quantity of interest ($$\overline{\Phi}$$)')
set(u,'Interpreter','latex','fontsize',12)
h = legend('$$\overline{E_{true}}$$','$$\overline{E}$$','$$\overline{E+D}$$','$$\overline{E+\alpha_{min}D}$$','$$\overline{E+\alpha_{run}D}$$','Location','Best')
set(h,'Interpreter','latex','fontsize',12)
print(gcf,'MinT_Specific_Averages.png','-dpng')
%General
plot(t_m,ones([Steps+1,1])*4.1803,t_m,[0;MinT_E_bar_run],t_m,[0;MinT_E_D_gen_t],'--',t_m,[0;MinT_E_D_gen_alpha_t],':',t_m,[0;MinT_E_D_gen_alpha_run_t],'-.');
xlabel('time (s)')
u = ylabel('Quantity of interest ($$\overline{\Phi}$$)')
set(u,'Interpreter','latex','fontsize',12)
h = legend('$$\overline{E_{true}}$$','$$\overline{E+D}$$','$$\overline{E+\alpha_{min}D}$$','$$\overline{E+\alpha_{run}D}$$','Location','Best')
set(h,'Interpreter','latex','fontsize',12)
print(gcf,'MinT_General_Averages.png','-dpng')
%Sine
plot(t_m,ones([Steps+1,1])*4.1803,t_m,[0;MinT_E_bar_run],t_m,[0;MinT_E_D_sine_t],'--',t_m,[0;MinT_E_D_sine_alpha_t],':',t_m,[0;MinT_E_D_sine_alpha_run_t],'-.');
xlabel('time (s)')
u = ylabel('Quantity of interest ($$\overline{\Phi}$$)')
set(u,'Interpreter','latex','fontsize',12)
h = legend('$$\overline{E_{true}}$$','$$\overline{E}$$','$$\overline{E+D}$$','$$\overline{E+\alpha_{min}D}$$','$$\overline{E+\alpha_{run}D}$$','Location','Best')
set(h,'Interpreter','latex','fontsize',12)
print(gcf,'MinT_Sine_Averages.png','-dpng')

%% Plot Side by Side Average Comparison
%Base Systems
plot(t_m,ones([Steps+1,1])*4.1803,t_m,[0;MinT_E_D_spec_t],'--',t_m,[0;MinT_E_D_gen_t],':',t_m,[0;MinT_E_D_sine_t],'-.');
xlabel('time (s)')
u = ylabel('Quantity of interest ($$\overline{\Phi}$$)')
set(u,'Interpreter','latex','fontsize',12)
legend('True','Base','Specific','General','Sine','Location','Best')
print(gcf,'MinT_Base_Averages.png','-dpng')

%alpha Systems
plot(t_m,ones([Steps+1,1])*4.1803,t_m,[0;MinT_E_D_spec_alpha_t],'--',t_m,[0;MinT_E_D_gen_alpha_t],':',t_m,[0;MinT_E_D_sine_alpha_t],'-.');
xlabel('time (s)')
u = ylabel('Quantity of interest ($$\overline{\Phi}$$)')
set(u,'Interpreter','latex','fontsize',12)
legend('True','Specific','General','Sine','Location','Best')
print(gcf,'MinT_Alpha_Averages.png','-dpng')

%alpha run Systems

plot(t_m,ones([Steps+1,1])*4.1803,t_m,[0;MinT_E_D_spec_alpha_run_t],'--',t_m,[0;MinT_E_D_gen_alpha_run_t],':',t_m,[0;MinT_E_D_sine_alpha_run_t],'-.');
```

```
xlabel('time (s)')
u = ylabel('Quantity of interest ($$\overline{\Phi}$$)')
legend('True','Specific','General','Sine','Location','Best')
set(u,'Interpreter','latex','fontsize',12)
print(gcf,'MinT_Alpha_Run_Averages.png','-dpng')

%% Plot All Variances

%Specific
plot(t_m,[0;MinT_Var],':',t_m,[0,MinT_Var_spec],'--',t_m,[0;MinT_Var_alpha_spec],'-.',t_m,[0;MinT_Var_alpha_run_spec]);
xlabel('time (s)')
ylabel('Variance (\sigma^2)')
legend('\sigma^2','\sigma_{spec}^2','\sigma_{spec alpha}^2','\sigma_{spec alpha run}^2','Location','Best')
print(gcf,'MinT_Specific_Variances.png','-dpng')
%General
plot(t_m,[0;MinT_Var],':',t_m,[0,MinT_Var_gen],'--',t_m,[0;MinT_Var_alpha_gen],'-.',t_m,[0;MinT_Var_alpha_run_gen]);
xlabel('time (s)')
ylabel('Variance (\sigma^2)')
legend('\sigma^2','\sigma_{gen}^2','\sigma_{gen alpha}^2','\sigma_{gen alpha run}^2','Location','Best')
print(gcf,'MinT_General_Variances.png','-dpng')
%Sine
plot(t_m,[0;MinT_Var],':',t_m,[0,MinT_Var_sine],'--',t_m,[0;MinT_Var_alpha_sine],'-.',t_m,[0;MinT_Var_alpha_run_sine]);
xlabel('time (s)')
ylabel('Variance (\sigma^2)')
legend('\sigma^2','\sigma_{sine}^2','\sigma_{sine alpha}^2','\sigma_{sine alpha run}^2','Location','Best')
print(gcf,'MinT_Sine_Variances.png','-dpng')



%% Plot Side by Side Variance Comparison
%Base
plot(t_m,[0;MinT_Var],':',t_m,[0,MinT_Var_spec],'--',t_m,[0,MinT_Var_gen],'-.',t_m,[0,MinT_Var_sine]);
xlabel('time (s)')
ylabel('Variance (\sigma^2)')
legend('\sigma^2','\sigma_{spec}^2','\sigma_{gen}^2','\sigma_{sine}^2','Location','Best')
print(gcf,'MinT_Base_Variances.png','-dpng')

%alpha
plot(t_m,[0;MinT_Var],':',t_m,[0;MinT_Var_alpha_spec],'--',t_m,[0;MinT_Var_alpha_gen],'-.',t_m,[0;MinT_Var_alpha_sine]);
xlabel('time (s)')
ylabel('Variance (\sigma^2)')
legend('\sigma^2','\sigma_{spec}^2','\sigma_{gen}^2','\sigma_{sine}^2','Location','Best')
print(gcf,'MinT_Alpha_Variances.png','-dpng')

%alpha run
plot(t_m,[0;MinT_Var],':',t_m,[0;MinT_Var_alpha_run_spec],'--',t_m,[0;MinT_Var_alpha_run_gen],'-.',t_m,[0;MinT_Var_alpha_run_sine]);
xlabel('time (s)')
ylabel('Variance (\sigma^2)')
legend('\sigma^2','\sigma_{spec}^2','\sigma_{gen}^2','\sigma_{sine}^2','Location
```

```
    ','Best')
print(gcf,'MinT_Alpha_Run_Variances_.png','-dpng')

%% Plot Side by Side alpha vs alpha run comparison
%all alphas

plot(t_m,[0;MinT_alpha_spec],':',t_m,[0;MinT_alpha_gen],'--',t_m,[0;
    MinT_alpha_sine],'-.');
xlabel('time (s)')
ylabel('\alpha')
legend('\alpha','\alpha_{spec}','\alpha_{gen}','\alpha_{sine}','Location','Best'
    )
print(gcf,'MinT_Alpha.png','-dpng')


%all alpha runs
plot(t_m,[0;MinT_alpha_spec_run],':',t_m,[0;MinT_alpha_gen_run],'--',t_m,[0;
    MinT_alpha_sine_run],'-.');
xlabel('time (s)')
ylabel('\alpha')
legend('\alpha_{spec}','\alpha_{gen}','\alpha_{sine}','Location','Best')
print(gcf,'MinT_Alpha_Run.png','-dpng')

%specs

plot(t_m,[0;MinT_alpha_spec],':',t_m,[0;MinT_alpha_spec_run],'--');
xlabel('time (s)')
ylabel('\alpha')
legend('\alpha_{spec}','\alpha_{spec run}','Location','Best')
print(gcf,'MinT_Alpha_Spec_.png','-dpng')

%gens
plot(t_m,[0;MinT_alpha_gen],':',t_m,[0;MinT_alpha_gen_run],'--');
xlabel('time (s)')
ylabel('\alpha')
legend('\alpha_{gen}','\alpha_{gen run}','Location','Best')
print(gcf,'MinT_Alpha_Gen_.png','-dpng')

%sines

plot(t_m,[0;MinT_alpha_sine],':',t_m,[0;MinT_alpha_sine_run],'--');
xlabel('time (s)')
ylabel('\alpha')
legend('\alpha_{sine}','\alpha_{sine run}','Location','Best')
print(gcf,'MinT_Alpha_Sine_.png','-dpng')
```