

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/342411465>

# The Acceleration of Infinite Time Averaging Methods for Chaotic Systems

Preprint · June 2020

DOI: 10.13140/RG.2.2.36675.63523/2

CITATIONS

0

READS

125

3 authors, including:



Jamell Ivan Samuels

N/A

21 PUBLICATIONS 0 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:



Original Papers [View project](#)



Riemann Hypothesis [View project](#)

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

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Background and Motivation . . . . .	1
1.2	Concept and Approach . . . . .	1
<b>2</b>	<b>Systems</b>	<b>2</b>
2.0.1	Case A . . . . .	2
2.0.2	Case B . . . . .	2
2.0.3	Case C . . . . .	3
2.0.4	Case E . . . . .	3
2.0.5	Case F . . . . .	4
<b>3</b>	<b>Theoretical Background</b>	<b>6</b>
<b>4</b>	<b>Case B Study</b>	<b>9</b>
<b>5</b>	<b>Results and Discussion</b>	<b>15</b>
5.0.1	Minimum Time . . . . .	16
5.0.2	Error Calculation . . . . .	17
<b>6</b>	<b>Conclusion</b>	<b>18</b>
<b>7</b>	<b>Appendices</b>	<b>20</b>
7.1	Standard Deviation Derivation . . . . .	20
7.2	Plots . . . . .	21
7.2.1	Case A . . . . .	21
7.2.2	Case C . . . . .	23
7.2.3	Case E . . . . .	24
7.2.4	Case F . . . . .	25
7.3	Code . . . . .	26
7.3.1	Initialising Code . . . . .	26
7.3.2	Base Code . . . . .	41
7.3.3	Specific Auxiliary . . . . .	45
7.3.4	General Auxiliary . . . . .	48
7.3.5	Sine Auxiliary . . . . .	51
7.3.6	Minimum Time Specific Auxiliary . . . . .	54
7.3.7	Minimum Time General Auxiliary . . . . .	57
7.3.8	Minimum Time Sine Auxiliary . . . . .	60
7.3.9	Plot Function . . . . .	63
7.3.10	Minimum Time Plot Function . . . . .	65

# List of Figures

1.1	Numerical solution of the Rössler attractor [5] . . . . .	1
2.1	Numerical solution of the Case A attractor . . . . .	2
2.2	Numerical solution of the Case B attractor . . . . .	3
2.3	Numerical solution of the Case C attractor . . . . .	4
2.4	Numerical solution of the Case E attractor . . . . .	4
2.5	Numerical solution of the Case F attractor . . . . .	5
4.1	Case - B Base Averages . . . . .	9
4.2	Case - B Minimum Time Base Averages . . . . .	9
4.3	Case - B $\alpha_{min}$ . . . . .	10
4.4	Case - B $\alpha_{run}$ . . . . .	10
4.5	Case - B $\alpha_{min}$ Averages . . . . .	10
4.6	Case - B $\alpha_{min}$ Variances . . . . .	10
4.7	Case - B $\alpha_{run}$ Averages . . . . .	11
4.8	Case - B $\alpha_{run}$ Variances . . . . .	11
4.9	Case - B General Auxiliary Averages . . . . .	12
4.10	Case - B Minimum Time General Auxiliary Averages . . . . .	12
4.11	Case - B General Auxiliary Variances . . . . .	12
4.12	Case - B Sine Auxiliary Averages . . . . .	12
4.13	Case - B Specific Auxiliary Averages . . . . .	13
4.14	Case - B Minimum Time Specific Auxiliary Averages . . . . .	13
4.15	Case - B Specific Auxiliary Variances . . . . .	13
4.16	Case - B Minimum Time Specific Auxiliary Variances . . . . .	13
4.17	Case - B Sine Auxiliary Averages . . . . .	14
4.18	Case - B Minimum Time Sine Auxiliary Averages . . . . .	14
4.19	Case - B Sine Auxiliary Variances . . . . .	14
4.20	Case - B Minimum Time Sine Auxiliary Averages . . . . .	14
7.1	Converging or Samuels Distribution . . . . .	20
7.2	Case - A Base Averages . . . . .	21
7.3	Case - A Minimum Time Base Averages . . . . .	21
7.4	Case - A $\alpha_{min}$ . . . . .	21
7.5	Case - A $\alpha_{run}$ . . . . .	21
7.6	Case - A $\alpha_{min}$ Averages . . . . .	22
7.7	Case - A $\alpha_{min}$ Variances . . . . .	22
7.8	Case - A $\alpha_{run}$ Averages . . . . .	22
7.9	Case - A $\alpha_{run}$ Variances . . . . .	22
7.10	Case - A General Auxiliary Averages . . . . .	23
7.11	Case - A Minimum Time General Auxiliary Averages . . . . .	23
7.12	Case - A General Auxiliary Variances . . . . .	23
7.13	Case - A Sine Auxiliary Averages . . . . .	23
7.14	Case - A Specific Auxiliary Averages . . . . .	24
7.15	Case - A Minimum Time Specific Auxiliary Averages . . . . .	24
7.16	Case - A Specific Auxiliary Variances . . . . .	24
7.17	Case - A Minimum Time Specific Auxiliary Variances . . . . .	24
7.18	Case - A Sine Auxiliary Averages . . . . .	25
7.19	Case - A Minimum Time Sine Auxiliary Averages . . . . .	25
7.20	Case - A Sine Auxiliary Variances . . . . .	25
7.21	Case - A Minimum Time Sine Auxiliary Averages . . . . .	25
7.22	Case - C Base Averages . . . . .	26
7.23	Case - C Minimum Time Base Averages . . . . .	26
7.24	Case - C $\alpha_{min}$ . . . . .	26
7.25	Case - C $\alpha_{run}$ . . . . .	26
7.26	Case - C $\alpha_{min}$ Averages . . . . .	26
7.27	Case - C $\alpha_{min}$ Variances . . . . .	26
7.28	Case - C $\alpha_{run}$ Averages . . . . .	27

7.29	Case - C $\alpha_{run}$ Variances . . . . .	27
7.30	Case - C General Auxiliary Averages . . . . .	27
7.31	Case - C Minimum Time General Auxiliary Averages . . . . .	27
7.32	Case - C General Auxiliary Variances . . . . .	27
7.33	Case - C Sine Auxiliary Averages . . . . .	27
7.34	Case - C Specific Auxiliary Averages . . . . .	28
7.35	Case - C Minimum Time Specific Auxiliary Averages . . . . .	28
7.36	Case - C Specific Auxiliary Variances . . . . .	28
7.37	Case - C Minimum Time Specific Auxiliary Variances . . . . .	28
7.38	Case - C Sine Auxiliary Averages . . . . .	29
7.39	Case - C Minimum Time Sine Auxiliary Averages . . . . .	29
7.40	Case - C Sine Auxiliary Variances . . . . .	29
7.41	Case - C Minimum Time Sine Auxiliary Averages . . . . .	29
7.42	Case - E Base Averages . . . . .	30
7.43	Case - E Minimum Time Base Averages . . . . .	30
7.44	Case - E $\alpha_{min}$ . . . . .	30
7.45	Case - E $\alpha_{run}$ . . . . .	30
7.46	Case - E $\alpha_{min}$ Averages . . . . .	30
7.47	Case - E $\alpha_{min}$ Variances . . . . .	30
7.48	Case - E $\alpha_{run}$ Averages . . . . .	31
7.49	Case - E $\alpha_{run}$ Variances . . . . .	31
7.50	Case - E General Auxiliary Averages . . . . .	31
7.51	Case - E Minimum Time General Auxiliary Averages . . . . .	31
7.52	Case - E General Auxiliary Variances . . . . .	31
7.53	Case - E Sine Auxiliary Averages . . . . .	31
7.54	Case - E Specific Auxiliary Averages . . . . .	32
7.55	Case - E Minimum Time Specific Auxiliary Averages . . . . .	32
7.56	Case - E Specific Auxiliary Variances . . . . .	32
7.57	Case - E Minimum Time Specific Auxiliary Variances . . . . .	32
7.58	Case - E Sine Auxiliary Averages . . . . .	33
7.59	Case - E Minimum Time Sine Auxiliary Averages . . . . .	33
7.60	Case - E Sine Auxiliary Variances . . . . .	33
7.61	Case - E Minimum Time Sine Auxiliary Averages . . . . .	33
7.62	Case - F Base Averages . . . . .	34
7.63	Case - F Minimum Time Base Averages . . . . .	34
7.64	Case - F $\alpha_{min}$ . . . . .	34
7.65	Case - F $\alpha_{run}$ . . . . .	34
7.66	Case - F $\alpha_{min}$ Averages . . . . .	34
7.67	Case - F $\alpha_{min}$ Variances . . . . .	34
7.68	Case - F $\alpha_{run}$ Averages . . . . .	35
7.69	Case - F $\alpha_{run}$ Variances . . . . .	35
7.70	Case - F General Auxiliary Averages . . . . .	35
7.71	Case - F Minimum Time General Auxiliary Averages . . . . .	35
7.72	Case - F General Auxiliary Variances . . . . .	35
7.73	Case - F Sine Auxiliary Averages . . . . .	35
7.74	Case - F Specific Auxiliary Averages . . . . .	36
7.75	Case - F Minimum Time Specific Auxiliary Averages . . . . .	36
7.76	Case - F Specific Auxiliary Variances . . . . .	36
7.77	Case - F Minimum Time Specific Auxiliary Variances . . . . .	36
7.78	Case - F Sine Auxiliary Averages . . . . .	37
7.79	Case - F Minimum Time Sine Auxiliary Averages . . . . .	37
7.80	Case - F Sine Auxiliary Variances . . . . .	37
7.81	Case - F Minimum Time Sine Auxiliary Averages . . . . .	37

# List of Tables

3.1	Auxiliary Functions . . . . .	6
5.1	Case Averages . . . . .	15
5.2	Case $\alpha_{min}$ Averages . . . . .	15
5.3	Case $\alpha_{run}$ Averages . . . . .	15
5.4	Base vs. Minimum Time Base Comparison . . . . .	16
5.5	$\alpha_{min}$ vs. Minimum Time $\alpha_{min}$ Comparison . . . . .	16
5.6	$\alpha_{run}$ vs. Minimum Time $\alpha_{run}$ Comparison . . . . .	16
5.7	$\alpha_{run}$ case errors . . . . .	17
5.8	Minimum Time $\alpha_{run}$ case errors . . . . .	17

### **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.

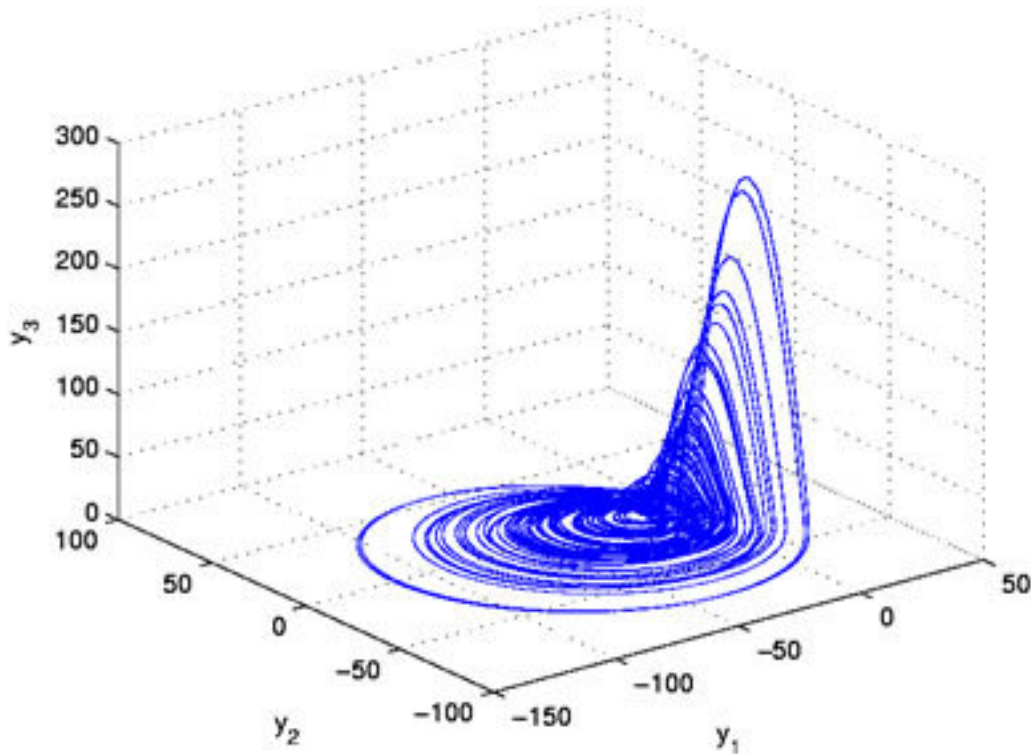


Figure 1.1: Numerical solution of the Rössler attractor [5]

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

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

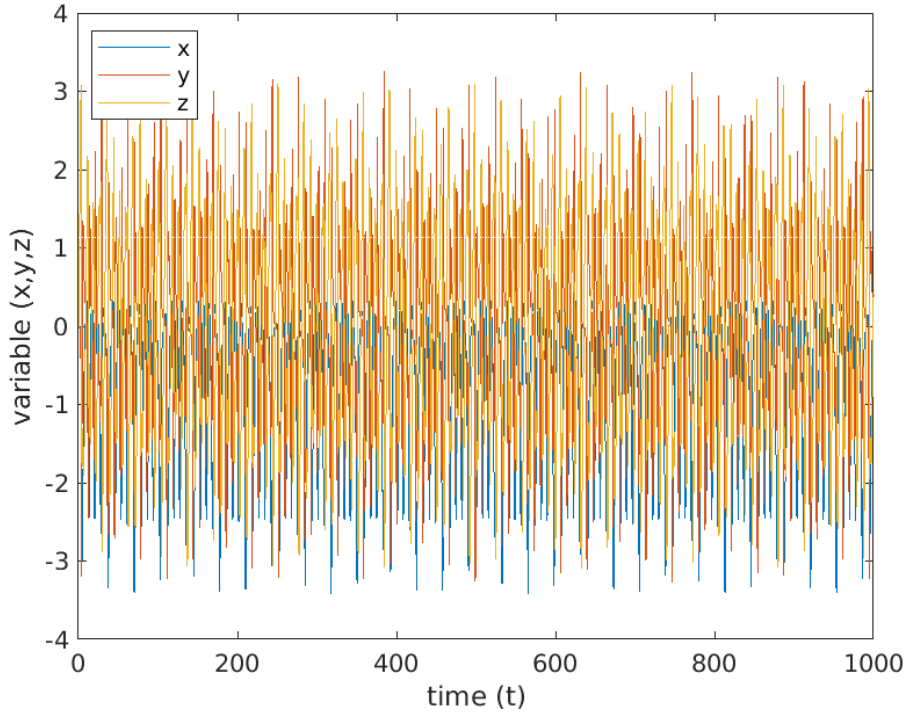


Figure 2.1: Numerical solution of the Case A attractor

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

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

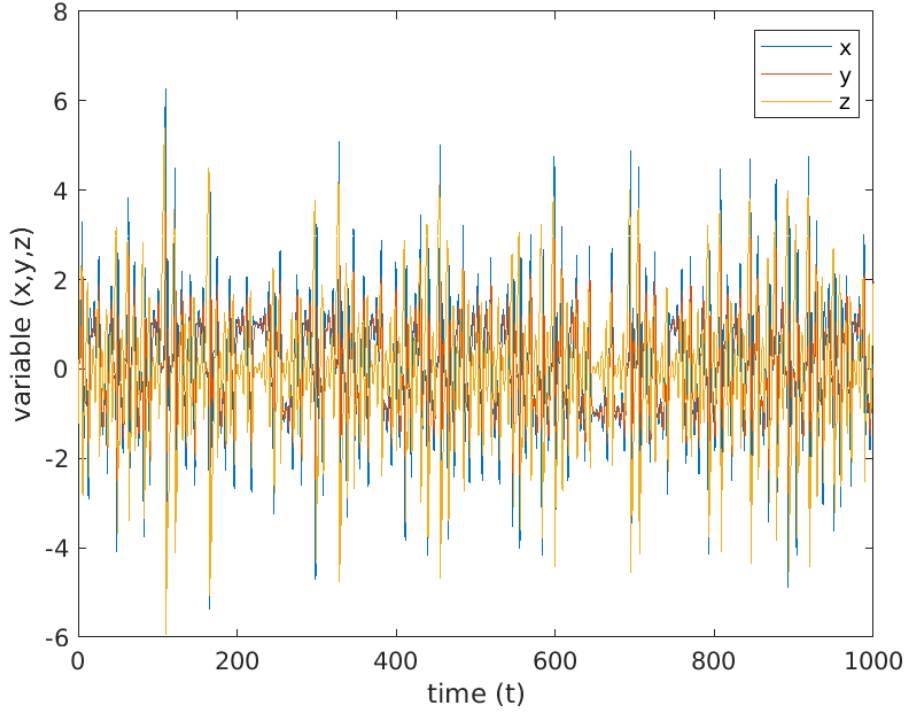


Figure 2.2: Numerical solution of the Case B attractor

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

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

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

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

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.

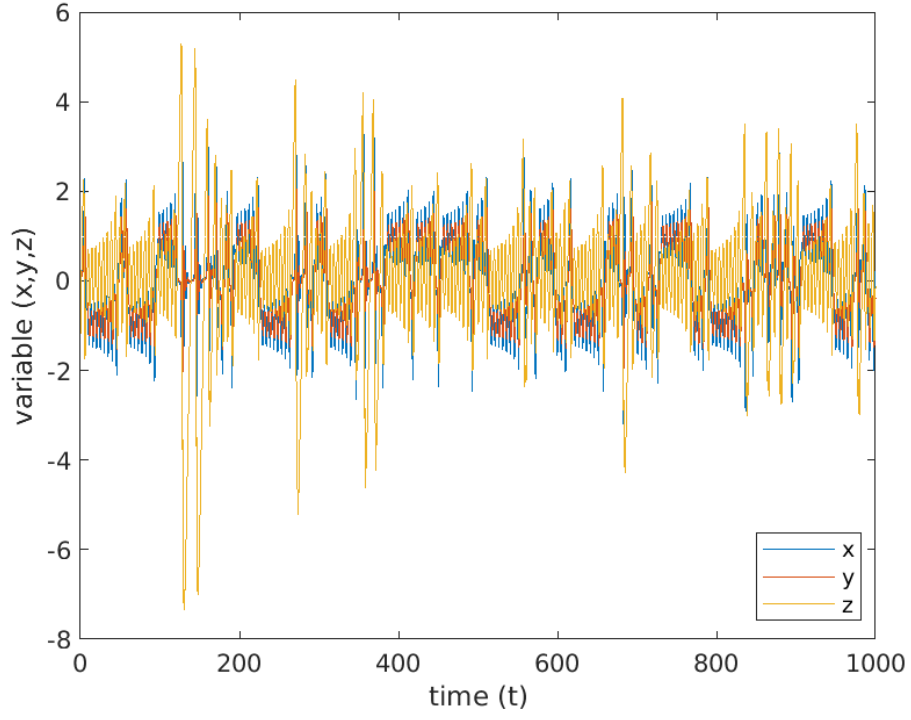


Figure 2.3: Numerical solution of the Case C attractor

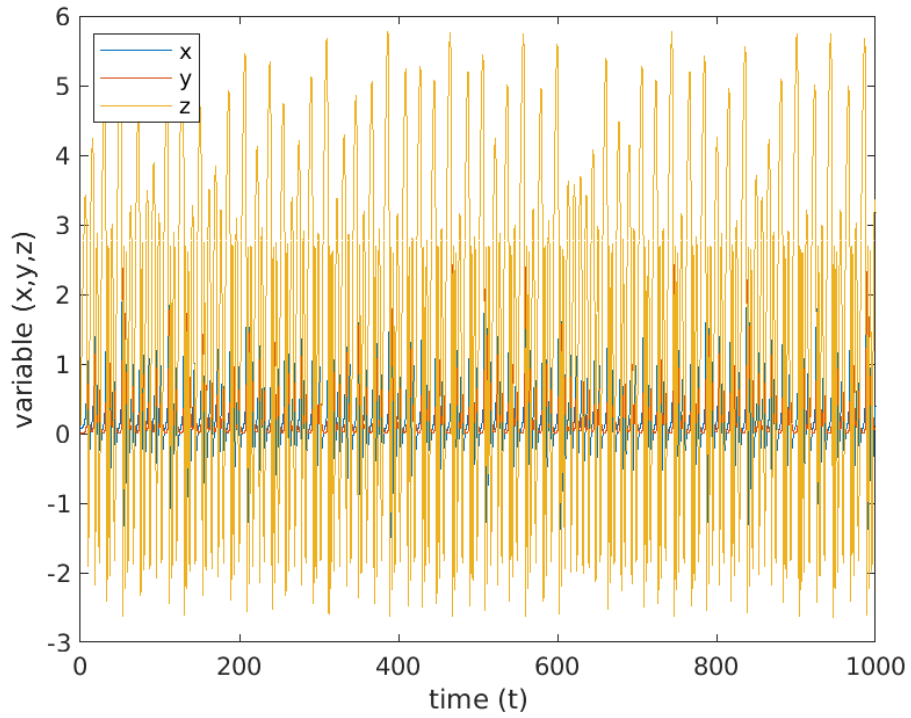


Figure 2.4: Numerical solution of the Case E attractor

### 2.0.5 Case F

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

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

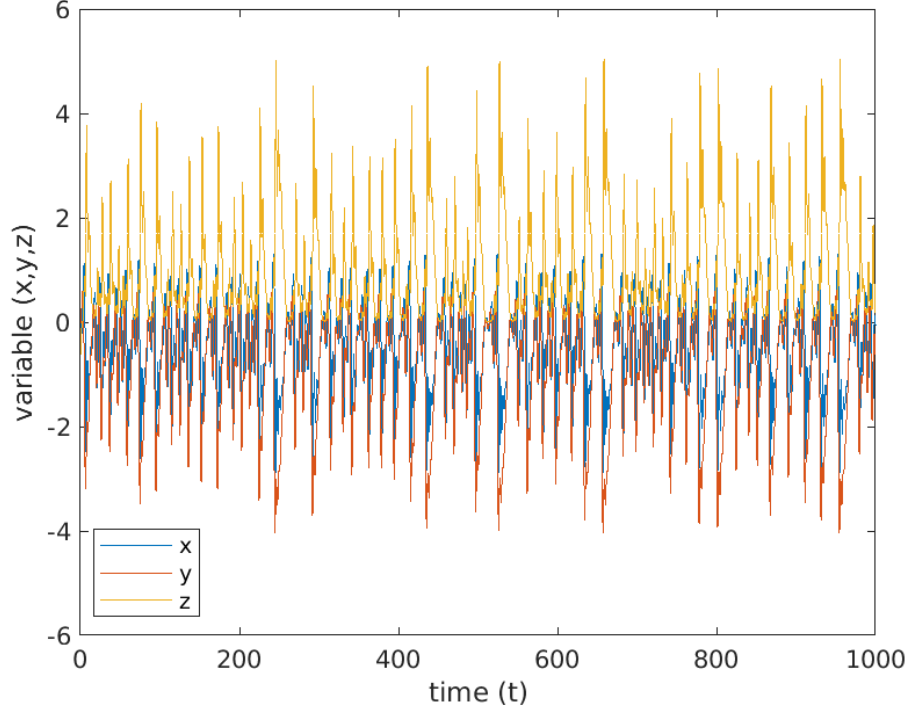


Figure 2.5: Numerical solution of the Case F attractor

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.

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

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.

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

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

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

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

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

Where  $D$  is the auxiliary function  $\frac{dV}{dt} = \frac{dV}{dx} \frac{dx}{dt}$ ,  $\alpha$  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.  
 $V(x) > 0 \forall x \in D$  and  $V(0) = 0$  i.e  $V(x)$  is positive semi definite and  $-\dot{V}(x) = -\frac{\Delta V}{\Delta x} f(x) \geq 0 \forall x \in \chi$ . 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  $\overline{E} + \overline{D}$  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	$V_1 = x$	$D_1 = \frac{dx}{dt}$
	$V_2 = y$	$D_2 = \frac{dy}{dt}$
	$V_3 = z$	$D_3 = \frac{dz}{dt}$
	$V_4 = xy$	$D_4 = x \frac{dx}{dt} + y \frac{dy}{dt}$
	$V_5 = xz$	$D_5 = x \frac{dz}{dt} + z \frac{dx}{dt}$
	$V_6 = yz$	$D_6 = y \frac{dz}{dt} + z \frac{dy}{dt}$
	$V_7 = xyz$	$D_7 = xy \frac{dz}{dt} + xz \frac{dy}{dt} + yz \frac{dx}{dt}$
	$V_8 = x^2$	$D_8 = 2x \frac{dx}{dt}$
	$V_9 = y^2$	$D_9 = 2y \frac{dy}{dt}$
	$V_{10} = z^2$	$D_{10} = 2z \frac{dz}{dt}$
Specific Auxiliary	$V_1 = x^2$	$D_1 = 2x \frac{dx}{dt}$
	$V_2 = xy$	$D_2 = x \frac{dy}{dt} + y \frac{dx}{dt}$
	$V_3 = xz$	$D_3 = x \frac{dz}{dt} + z \frac{dx}{dt}$
	$V_4 = y^2$	$D_4 = 2y \frac{dy}{dt}$
	$V_5 = yz$	$D_5 = 2y \frac{dz}{dt} + z \frac{dy}{dt}$
	$V_6 = z^2$	$D_6 = 2z \frac{dz}{dt}$
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  $\phi$  in comparison to  $E$ . The variance of  $E$  and  $\phi$  are defined such that.

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

$$\sigma_\phi^2 = \overline{(E + \alpha_i D_i + \bar{E} + \alpha_i \bar{D}_i)^2}. \quad (3.6)$$

Which can be further expanded to.

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

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

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

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

The expression for  $\alpha_{min}$  then becomes.

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

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

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

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.

$$\omega = [\omega_x, \omega_y, \omega_z] \quad (3.9)$$

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

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

Ergo 0 angular and 0 linear net momentum.

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

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  $t \frac{d^2\bar{x}}{dt^2} < (t)_{E_{minimum}}$  and  $\frac{d^2\bar{x}}{dt^2} t_2 > E(t)_{maximum}$ . where  $t_1$  and  $t_2$  are both stationary points and  $t$  of  $E(t)_{maximum}$  and  $t$  of  $E(t)_{minimum}$  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  $\mu$  or  $\bar{E}$ . This is calculated as  $s_d = (x_i - \bar{x})$ . By calculating the covariance between the sample deviation and the average, a relationship is established whereby it can be stated that the average at  $T \rightarrow \infty$  for any system is located where  $s_d = 0$ .

The derivation for this follows.

$$T \rightarrow \infty \text{ Cov}(s_d, \bar{E}) \rightarrow 0 \quad (3.11)$$

$$\frac{1}{N} \Sigma(\overline{E_T} - \overline{E_\infty})(s_d - \bar{s}_d) \rightarrow 0 \quad (3.12)$$

$$\begin{aligned}
T &\Rightarrow \infty \quad \overline{s_d} \rightarrow 0 \\
\frac{1}{N}(\Sigma(\overline{E_T} - \overline{E_\infty}))(s_d) &\rightarrow 0 \\
\Sigma \frac{d\overline{E_T}}{dt} s_d + \Sigma \overline{E_T} \frac{ds_d}{dt} - \Sigma \frac{d\overline{E_\infty}}{dt} - \Sigma \frac{ds_d}{dt} \overline{E_{infy}} &\rightarrow 0 \\
\frac{ds_d}{dt} = 0, \quad \frac{d\overline{E_\infty}}{dt} &= 0 \\
\Sigma \frac{dE}{dt} s_d &\rightarrow 0 \\
\Sigma_1^N s_d &\rightarrow 0.
\end{aligned}$$

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.

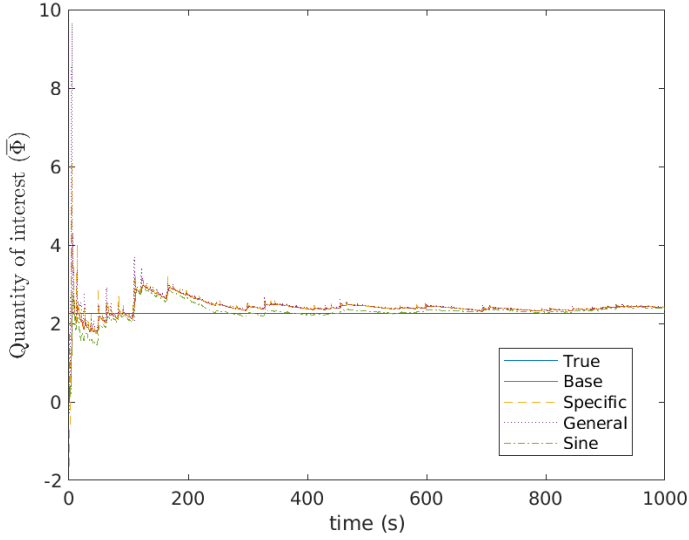


Figure 4.1: Case - B Base Averages

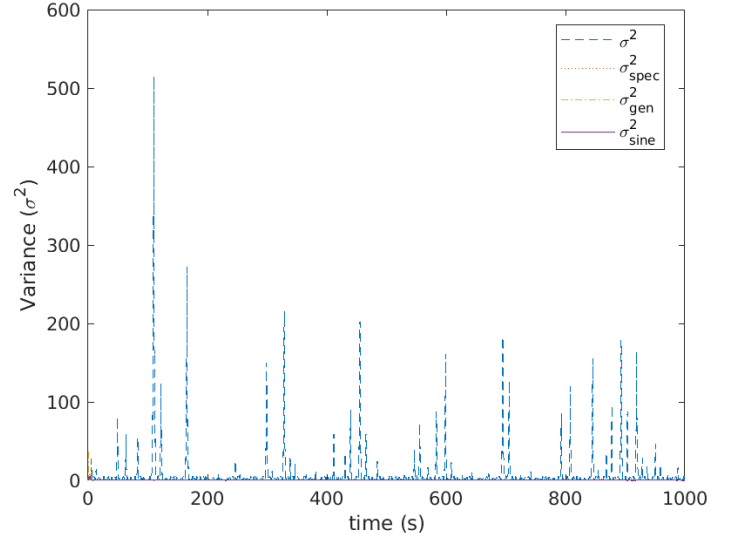


Figure 4.2: Case - B Minimum Time Base Averages

The first study shows the evolution of the base  $\bar{E}$  alongside the calculated  $\bar{\phi}$ '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.

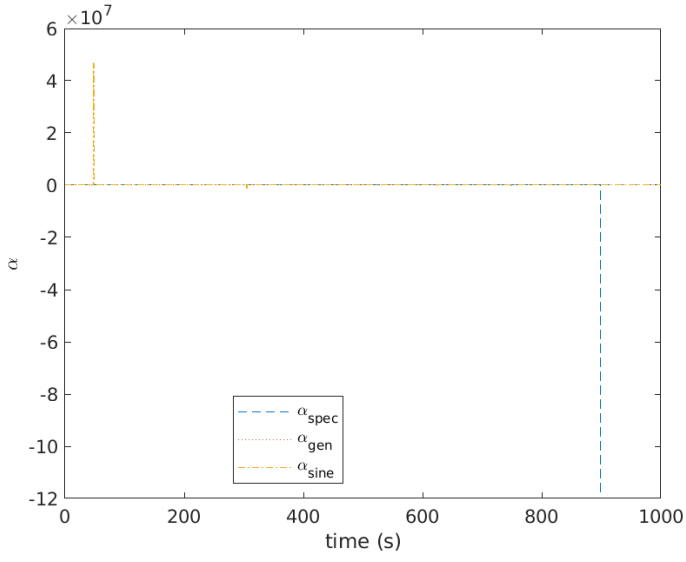


Figure 4.3: Case - B  $\alpha_{min}$

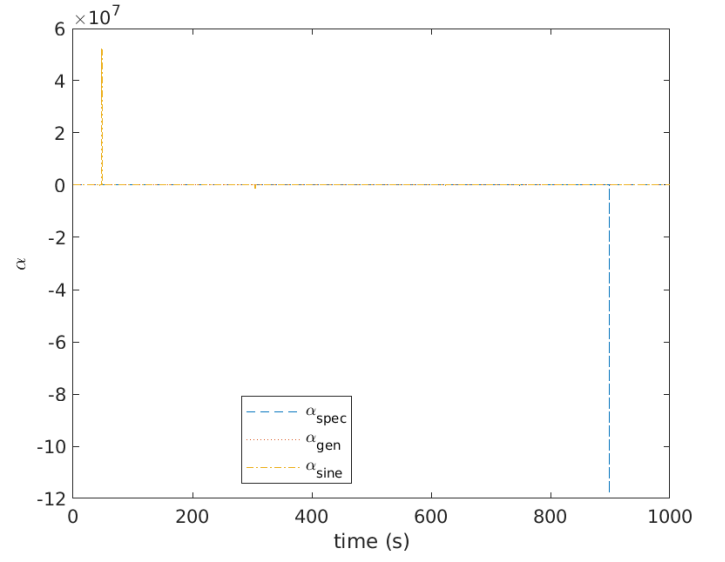


Figure 4.4: Case - B  $\alpha_{run}$

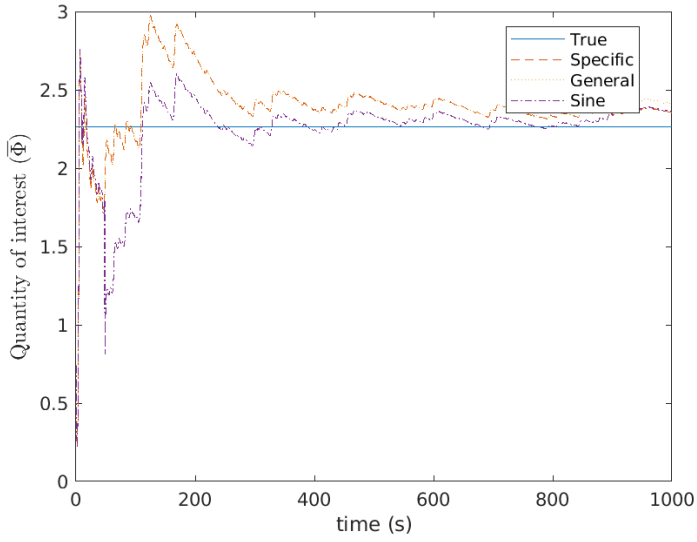


Figure 4.5: Case - B  $\alpha_{min}$  Averages

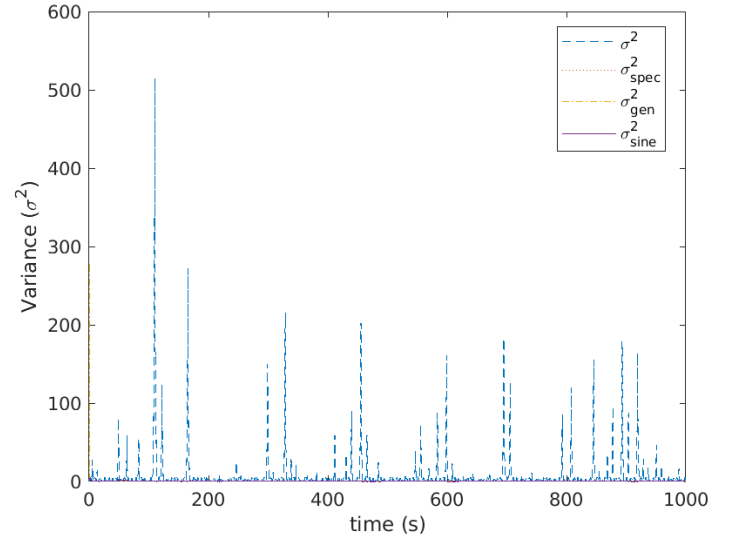


Figure 4.6: Case - B  $\alpha_{min}$  Variances

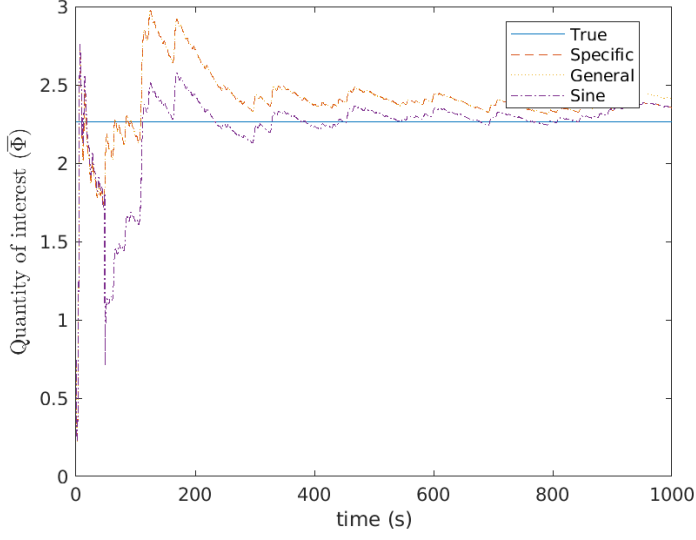


Figure 4.7: Case - B  $\alpha_{run}$  Averages

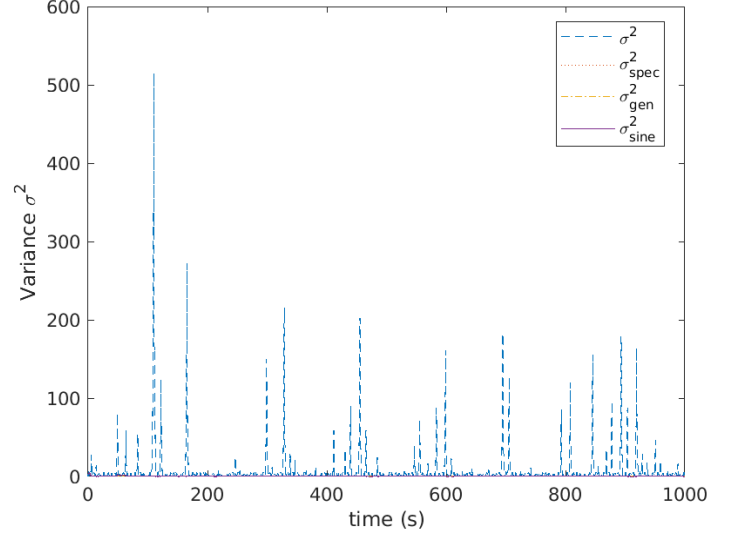


Figure 4.8: Case - B  $\alpha_{run}$  Variances

Graphically, the  $\alpha_{min}$  and  $\alpha_{run}$  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  $\alpha$  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  $\alpha$ .

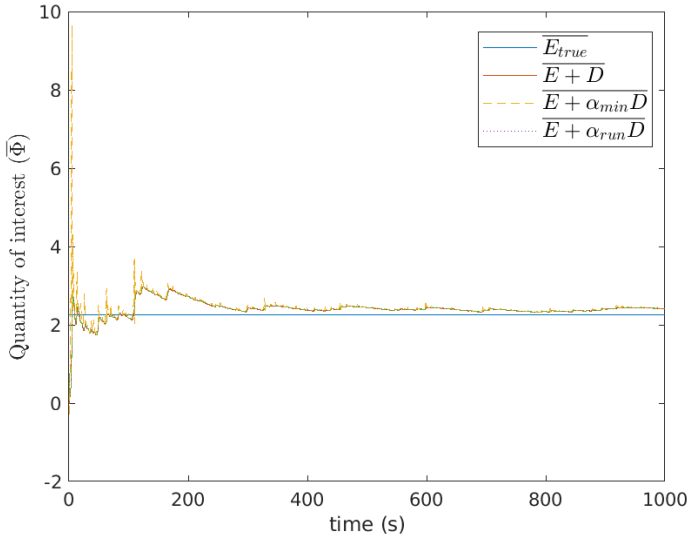


Figure 4.9: Case - B General Auxiliary Averages

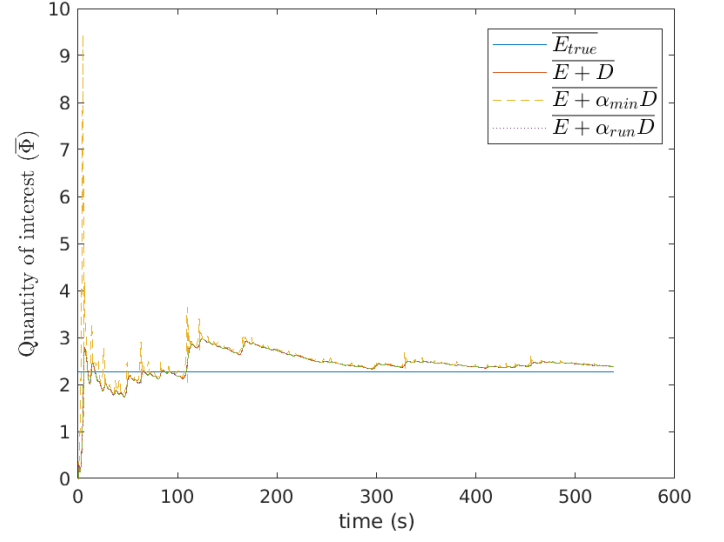


Figure 4.10: Case - B Minimum Time General Auxiliary Averages

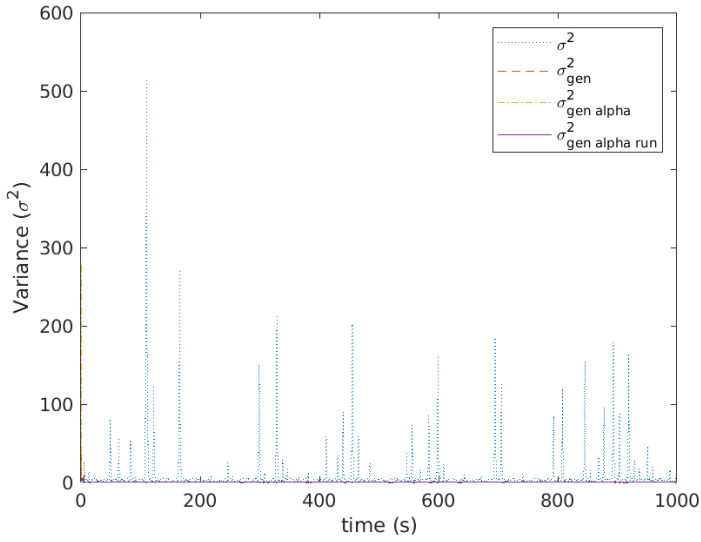


Figure 4.11: Case - B General Auxiliary Variances

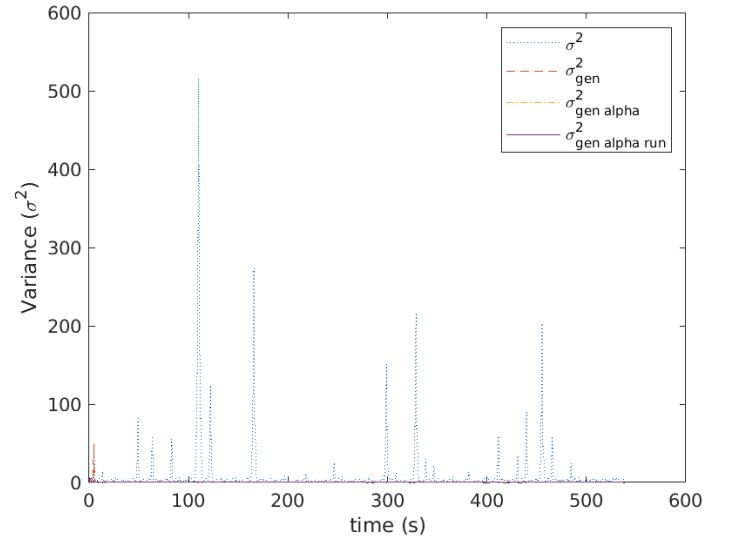


Figure 4.12: Case - B Sine Auxiliary Averages

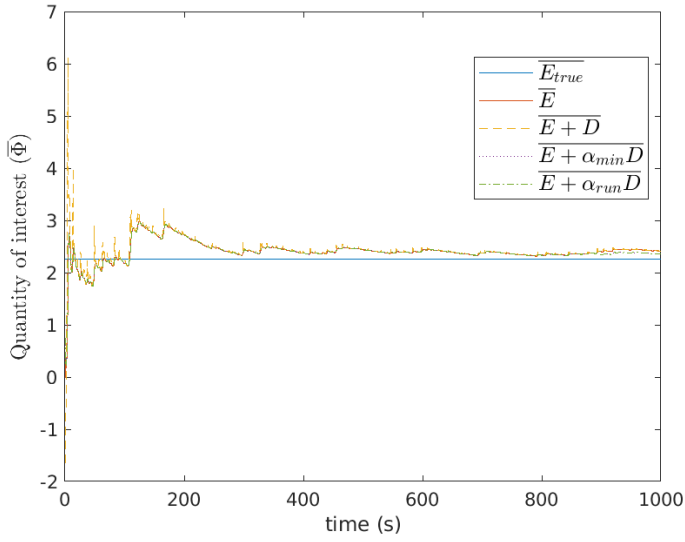


Figure 4.13: Case - B Specific Auxiliary Averages

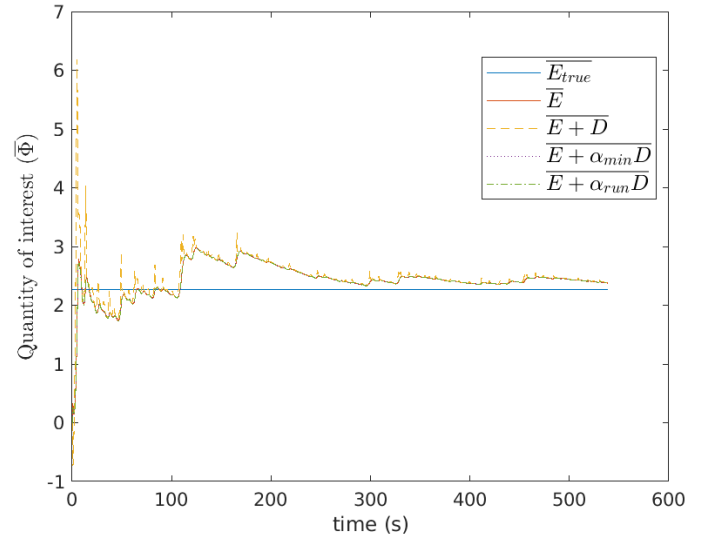


Figure 4.14: Case - B Minimum Time Specific Auxiliary Averages

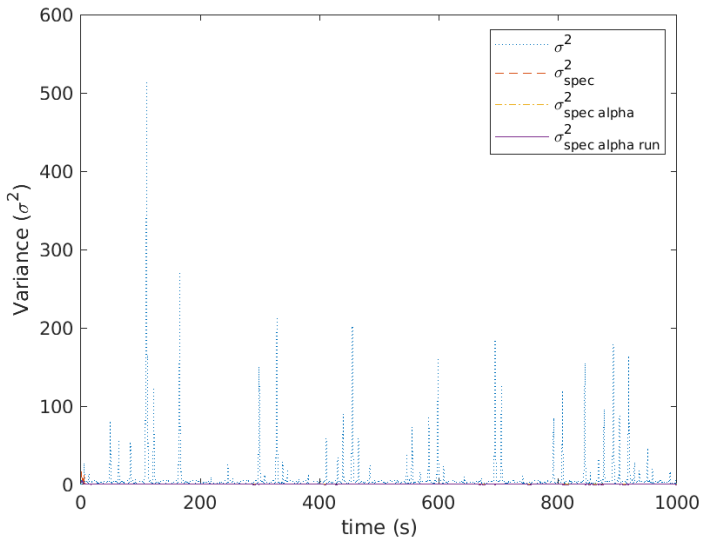


Figure 4.15: Case - B Specific Auxiliary Variances

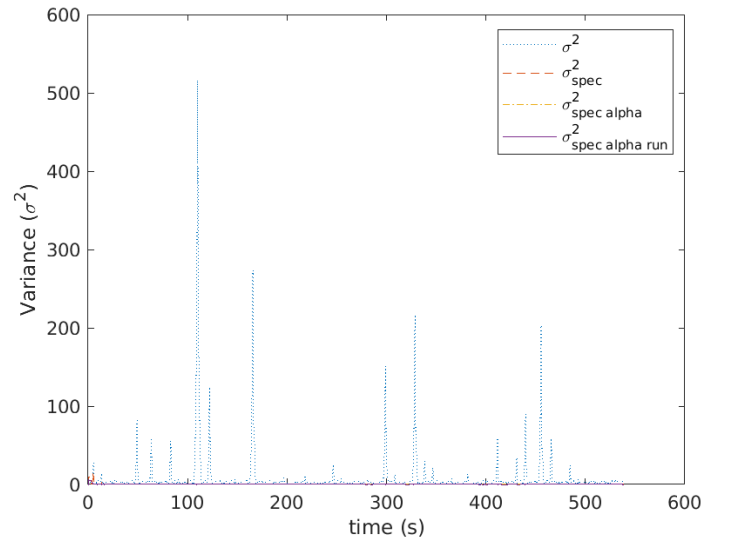


Figure 4.16: Case - B Minimum Time Specific Auxiliary Variances

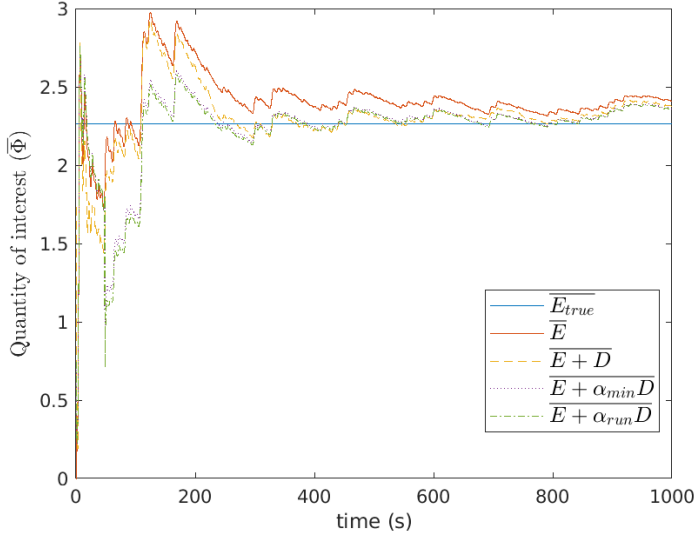


Figure 4.17: Case - B Sine Auxiliary Averages

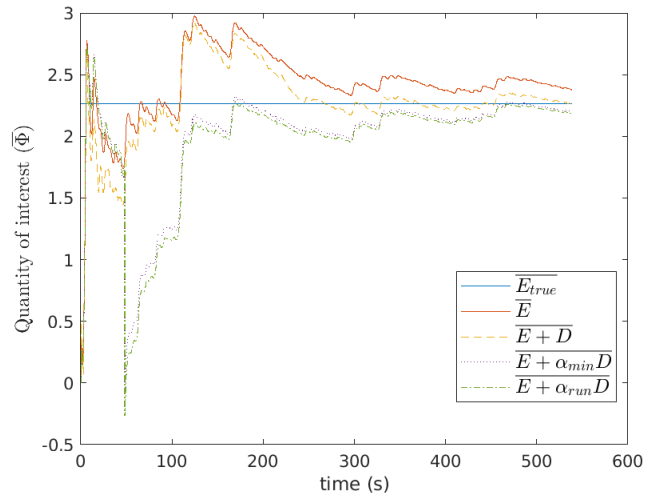


Figure 4.18: Case - B Minimum Time Sine Auxiliary Averages

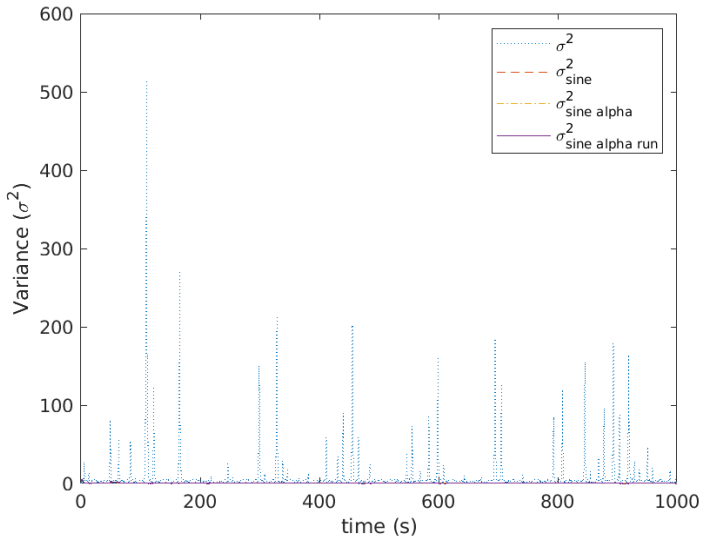


Figure 4.19: Case - B Sine Auxiliary Variances

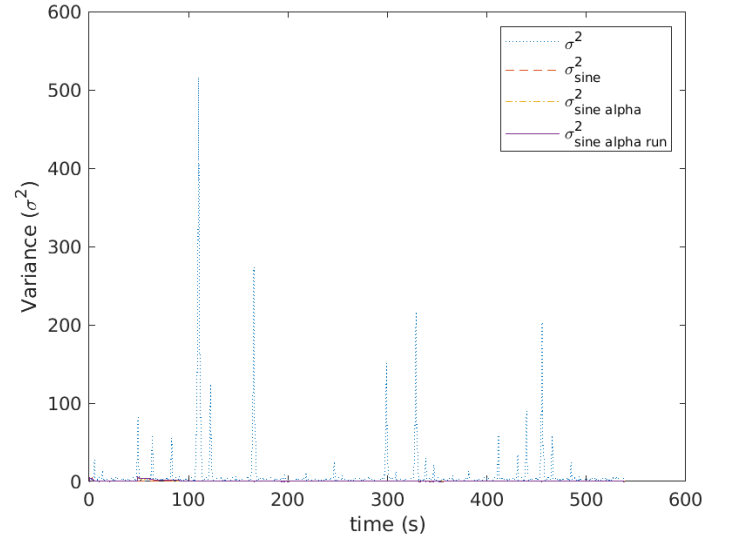


Figure 4.20: Case - B Minimum Time Sine Auxiliary Averages

The minimum time cases demonstrate slightly more erratic behaviour before settling. It can be seen that the use of an  $\alpha$  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 then 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  $D_{spec}$  however  $D_{sine}$  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  $\alpha_{min}$  Averages

After multiplying the auxiliary function by  $\alpha_{min}$ ,  $E_{sd}$  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  $\alpha$  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  $\alpha_{run}$  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	$\overline{E_{T=100000}}$	$\overline{E + D_{gen}}$	$\overline{E + D_{spec}}$	$\overline{E + D_{sine}}$	$\overline{E_{minT}}$	$\overline{E + D_{gen_{minT}}}$	$\overline{E + D_{spec_{minT}}}$	$\overline{E + D_{sine_{minT}}}$
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	$\overline{E_{T=100000}}$	$\overline{E + \alpha D_{gen}}$	$\overline{E + \alpha D_{spec}}$	$\overline{E + \alpha D_{sine}}$	$\overline{E_{minT}}$	$\overline{E + \alpha D_{gen_{minT}}}$	$\overline{E + \alpha D_{spec_{minT}}}$	$\overline{E + \alpha D_{sine_{minT}}}$
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:  $\alpha_{min}$  vs. Minimum Time  $\alpha_{min}$  Comparison

Case	$\overline{E_{T=100000}}$	$\overline{E + \alpha D_{gen}}$	$\overline{E + \alpha D_{spec}}$	$\overline{E + \alpha D_{sine}}$	$\overline{E_{minT}}$	$\overline{E + \alpha D_{gen_{minT}}}$	$\overline{E + \alpha D_{spec_{minT}}}$	$\overline{E + \alpha D_{sine_{minT}}}$
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:  $\alpha_{run}$  vs. Minimum Time  $\alpha_{run}$  Comparison



### 5.0.2 Error Calculation

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

Case	$\bar{E}$	$\overline{E + \alpha D_{gen}}$	$\overline{E + \alpha D_{spec}}$	$\overline{E + \alpha D_{sine}}$	$\overline{E_{sd}}$
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:  $\alpha_{run}$  case errors

Case	$\bar{E}_{minT}$	$\overline{E + \alpha D_{gen_{minT}}}$	$\overline{E + \alpha D_{spec_{minT}}}$	$\overline{E + \alpha D_{sine_{minT}}}$
A	0.1257	$8.877 \times 10^{-5}$	$6.658 \times 10^{-4}$	$-6.2145 \times 10^{-4}$
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  $\alpha_{run}$  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

$$\begin{aligned}
 T \rightarrow \infty \quad \text{Cov}(\sigma^2, \bar{E}_T) &\rightarrow 0 \\
 \frac{1}{N-1} \Sigma (\bar{E}_T - \bar{E}_\infty)(\sigma_T^2 - \bar{\sigma}_T^2) &\rightarrow 0 \\
 \bar{\sigma}_T^2 &= 0 \quad \forall T \\
 \Sigma (\bar{E}_T - \bar{E}_\infty) \sigma_T^2 &\rightarrow 0 \\
 \Sigma \bar{E}_T \frac{d\sigma_T^2}{dt} + \Sigma \sigma_T^2 \frac{d\bar{E}_T}{dt} - \Sigma \bar{E}_\infty \frac{d\sigma_T^2}{dt} - \Sigma \sigma_T^2 \frac{d\bar{E}_\infty}{dt} &\rightarrow 0 \\
 T \rightarrow \infty \quad \frac{d\bar{E}_\infty}{dt} = 0 \quad \frac{d\sigma_T^2}{dt} &= 0
 \end{aligned}$$

variance in time does not change for a chaotic system.

$\therefore$

$$\Sigma \sigma_T^2 \frac{d\bar{E}_T}{dt} \rightarrow 0$$

Of note.

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

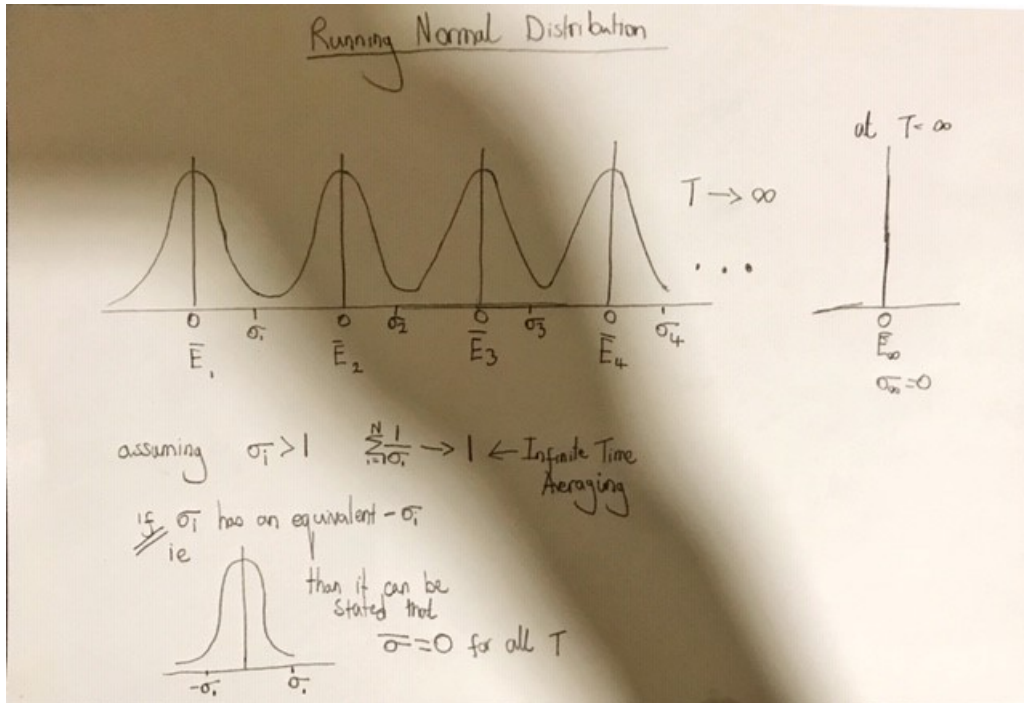


Figure 7.1: Converging Distribution

correction - there's a small mistake regarding the convergence of  $\sigma$  to 1. It converges to  $\infty$ . regardless it doesn't matter.

## 7.2 Plots

### 7.2.1 Case A

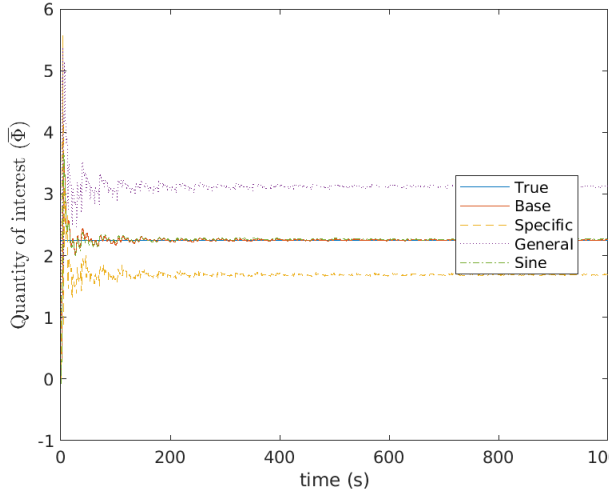


Figure 7.2: Case - A Base Averages

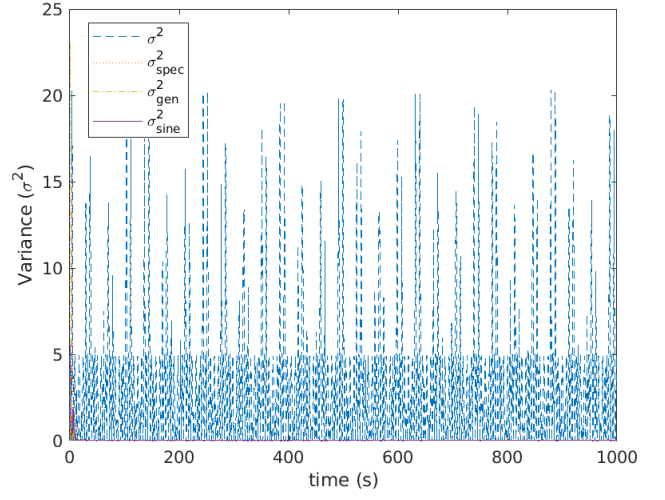


Figure 7.3: Case - A Minimum Time Base Averages

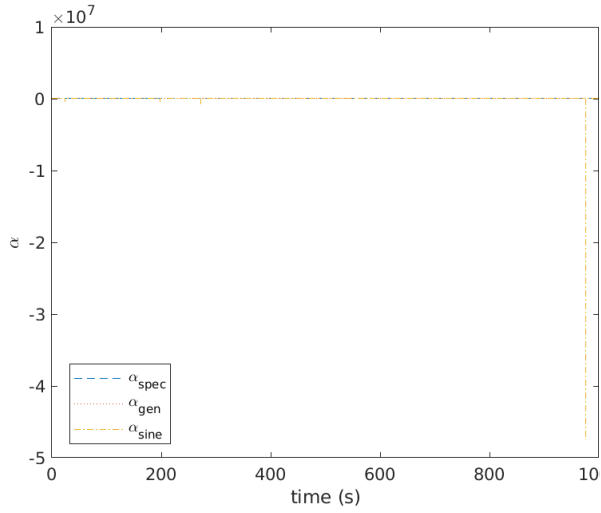


Figure 7.4: Case - A  $\alpha_{min}$

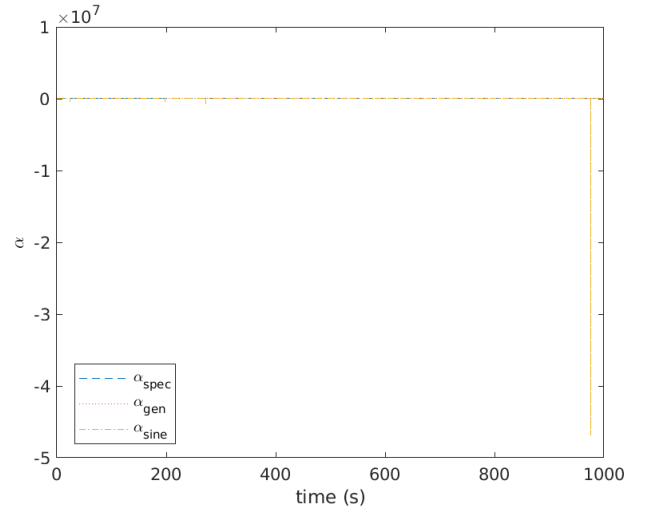


Figure 7.5: Case - A  $\alpha_{run}$

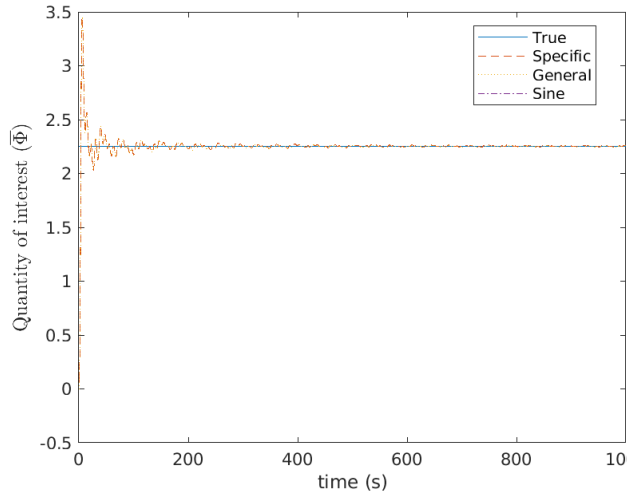


Figure 7.6: Case - A  $\alpha_{min}$  Averages

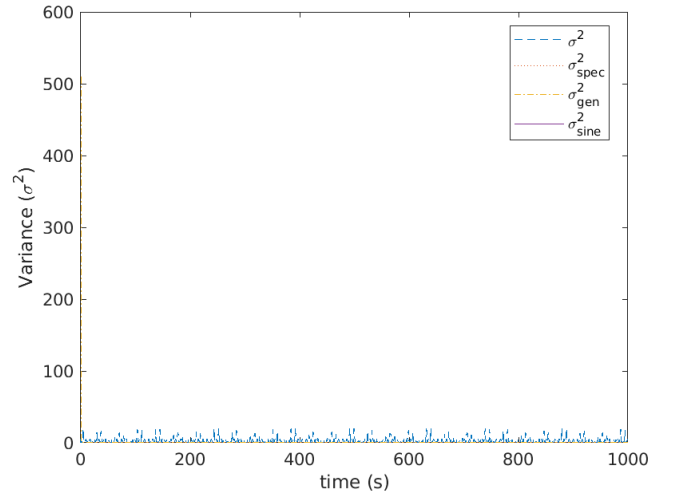


Figure 7.7: Case - A  $\alpha_{min}$  Variances

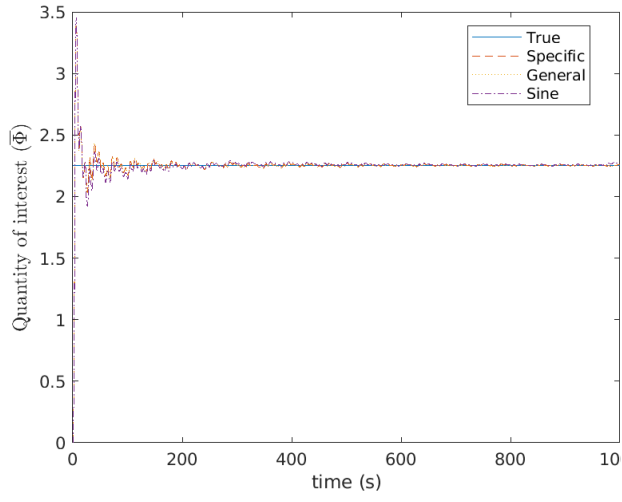


Figure 7.8: Case - A  $\alpha_{run}$  Averages

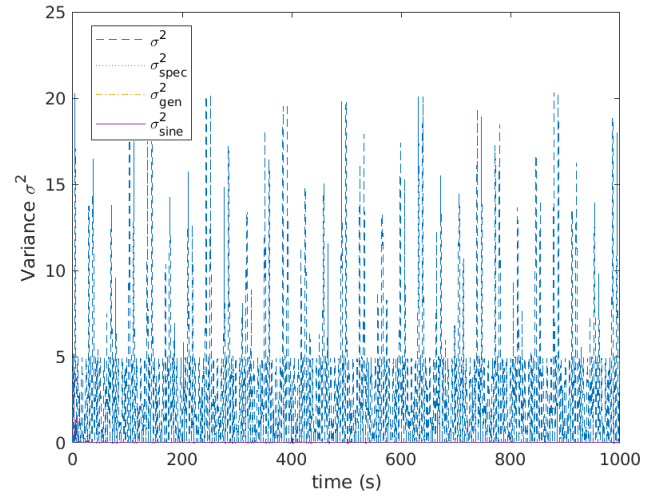


Figure 7.9: Case - A  $\alpha_{run}$  Variances

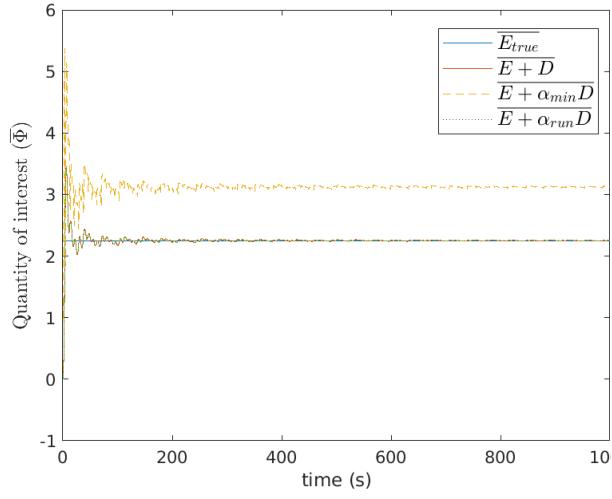


Figure 7.10: Case - A General Auxiliary Averages

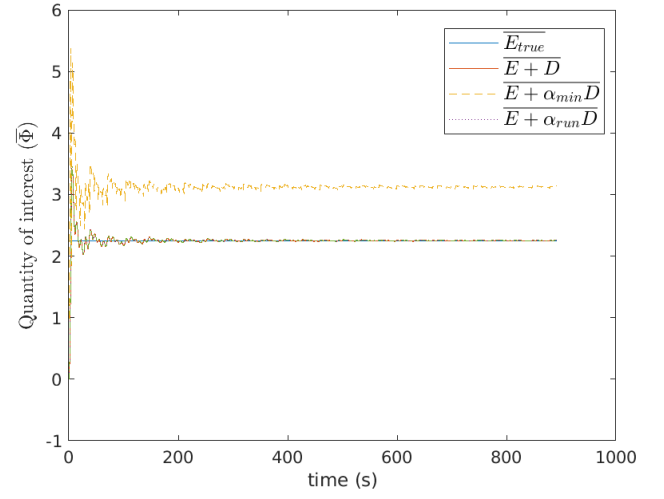


Figure 7.11: Case - A Minimum Time General Auxiliary Averages

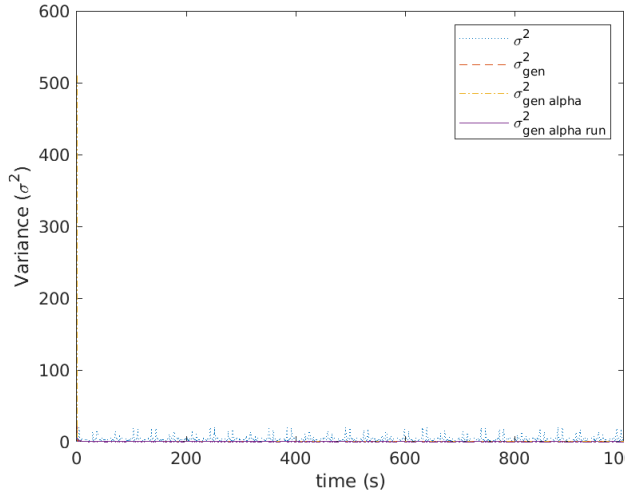


Figure 7.12: Case - A General Auxiliary Variances

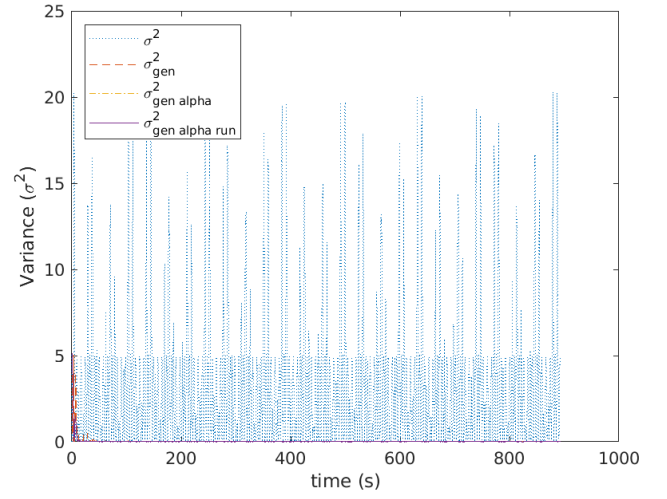


Figure 7.13: Case - A Sine Auxiliary Averages

## 7.2.2 Case C

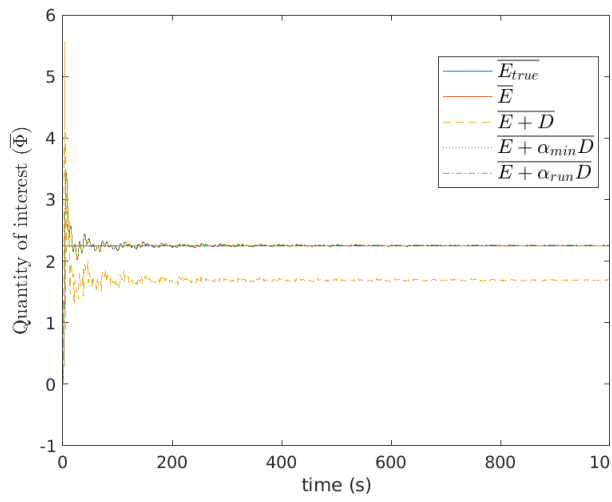


Figure 7.14: Case - A Specific Auxiliary Averages

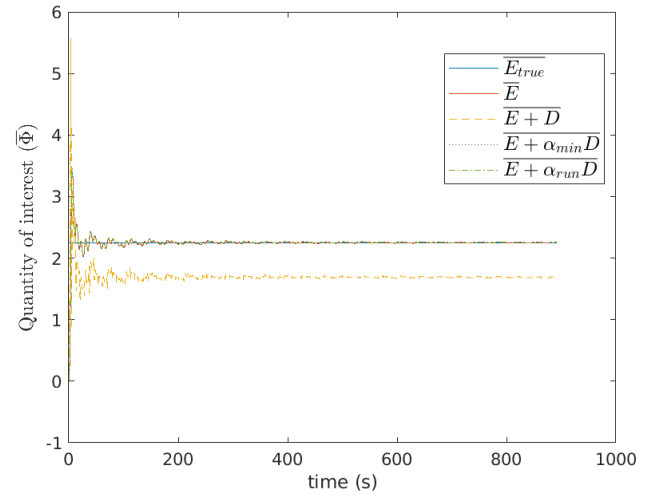


Figure 7.15: Case - A Minimum Time Specific Auxiliary Averages

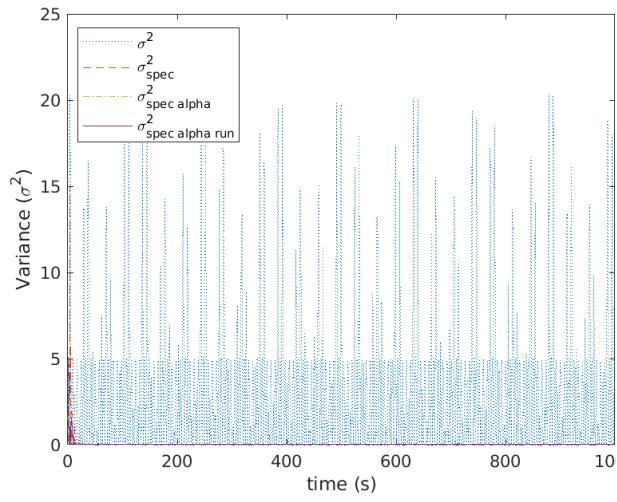


Figure 7.16: Case - A Specific Auxiliary Variances

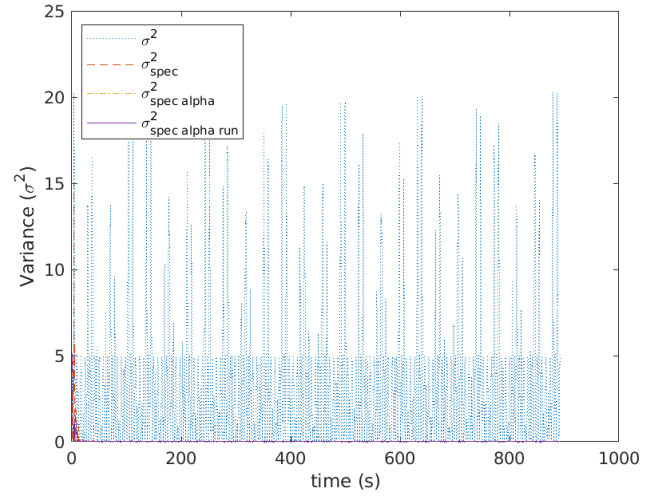


Figure 7.17: Case - A Minimum Time Specific Auxiliary Variances

### 7.2.3 Case E



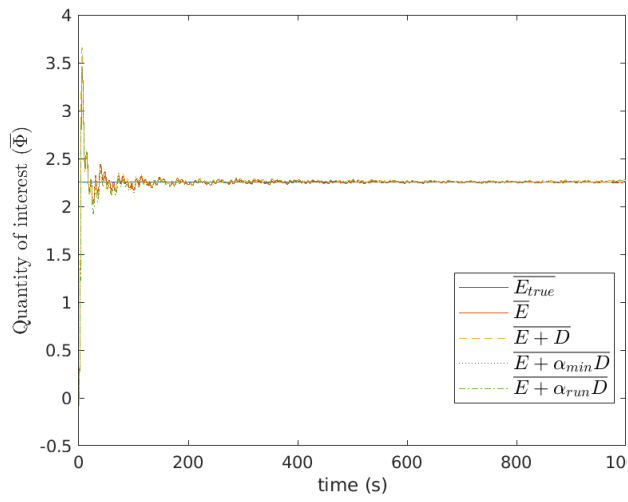


Figure 7.18: Case - A Sine Auxiliary Averages

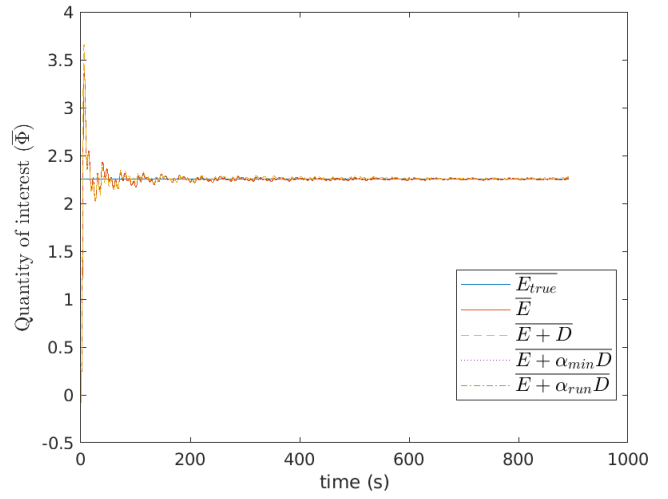


Figure 7.19: Case - A Minimum Time Sine Auxiliary Averages

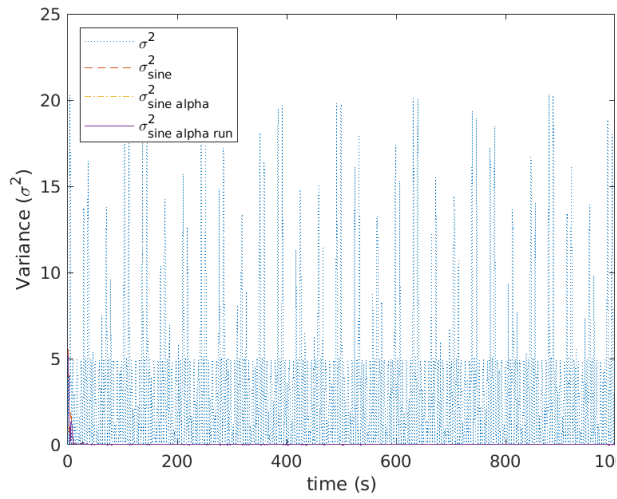


Figure 7.20: Case - A Sine Auxiliary Variances

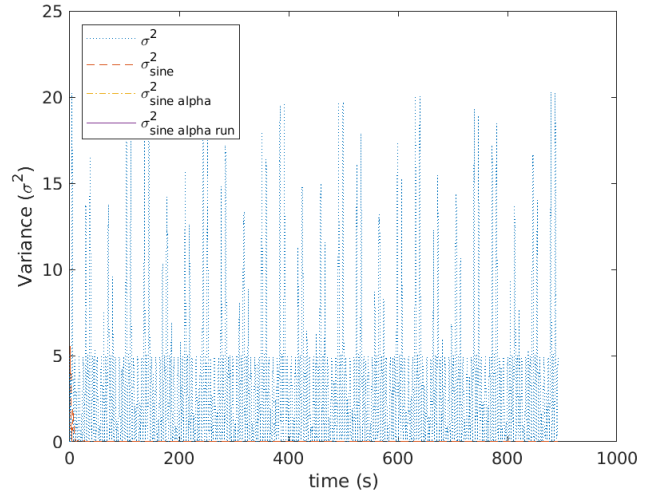


Figure 7.21: Case - A Minimum Time Sine Auxiliary Averages

## 7.2.4 Case F



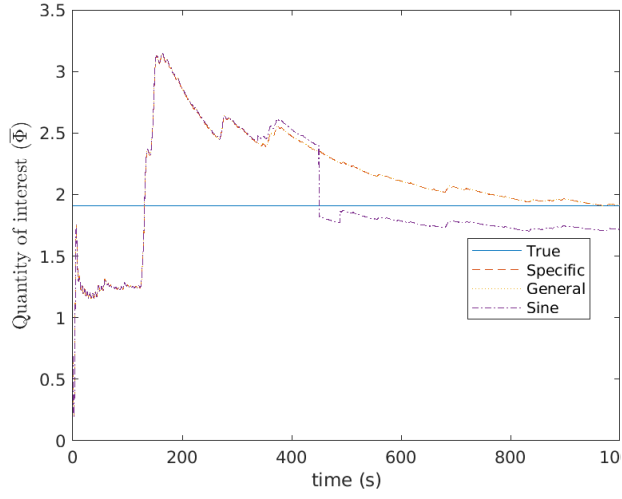


Figure 7.28: Case - C  $\alpha_{run}$  Averages

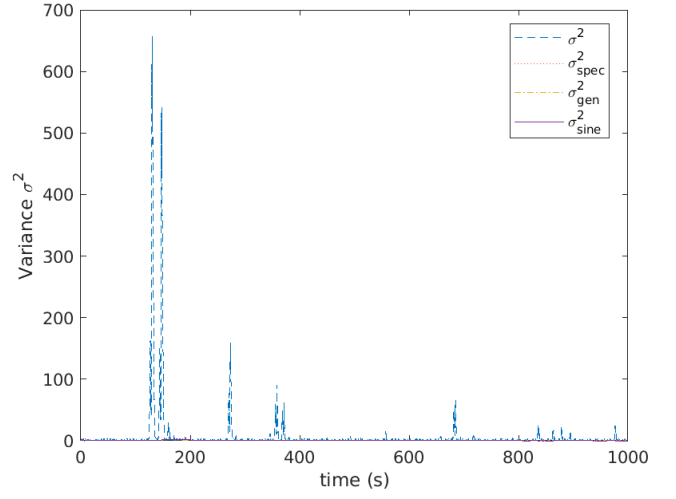


Figure 7.29: Case - C  $\alpha_{run}$  Variances

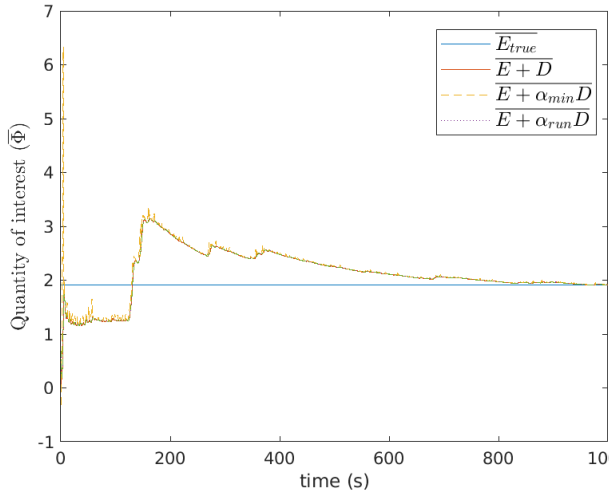


Figure 7.30: Case - C General Auxiliary Averages

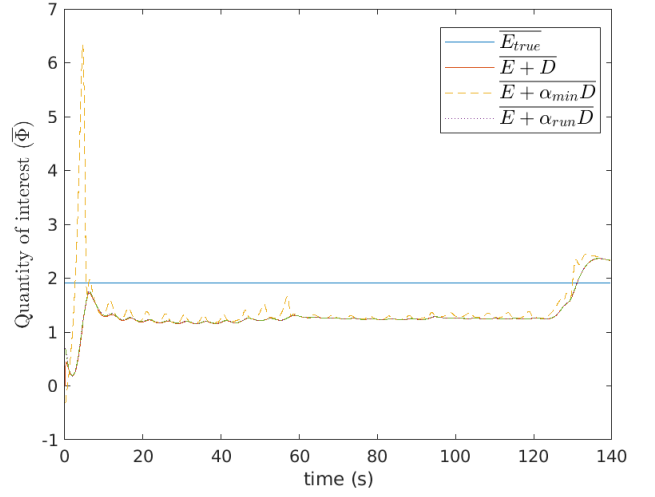


Figure 7.31: Case - C Minimum Time General Auxiliary Averages

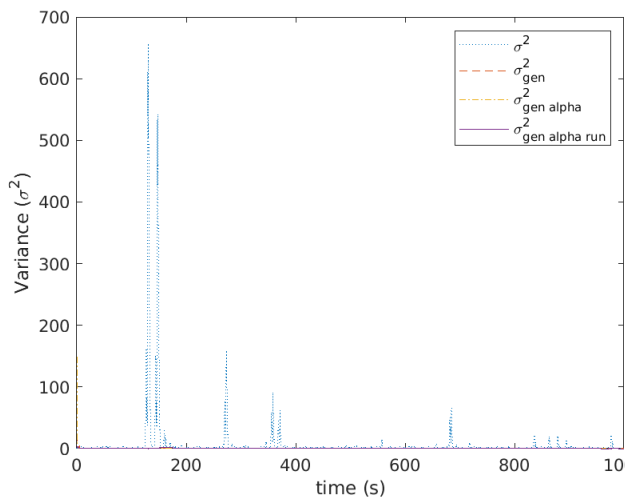


Figure 7.32: Case - C General Auxiliary Variances

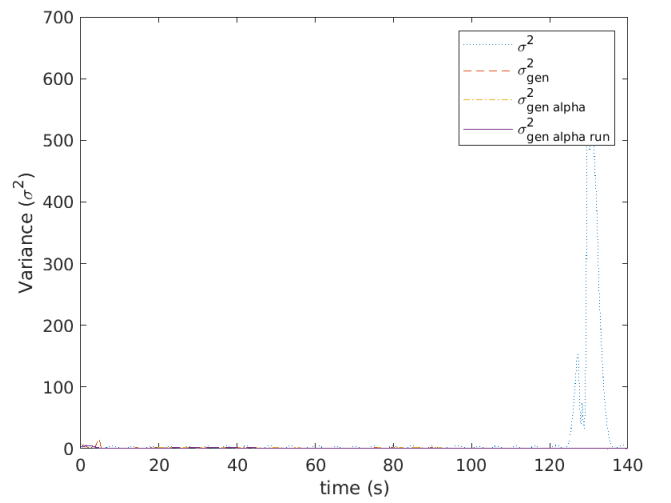


Figure 7.33: Case - C Sine Auxiliary Averages

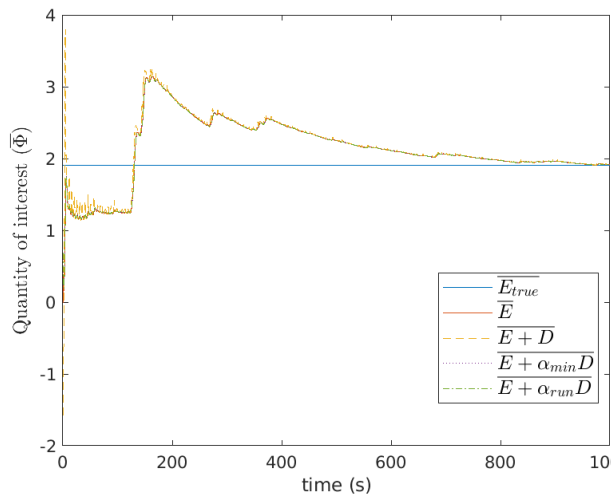


Figure 7.34: Case - C Specific Auxiliary Averages

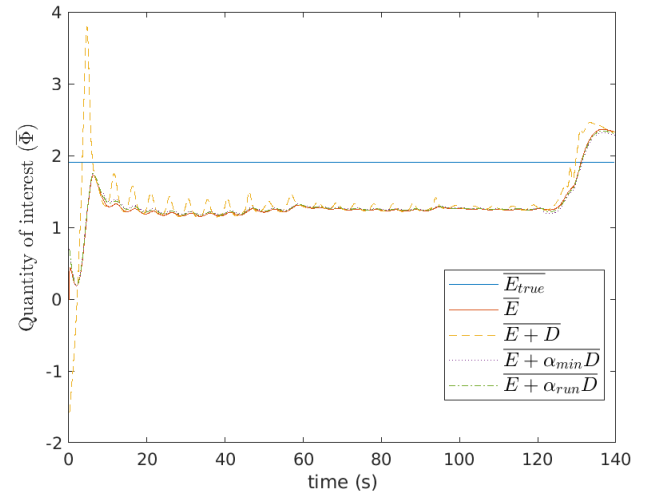


Figure 7.35: Case - C Minimum Time Specific Auxiliary Averages

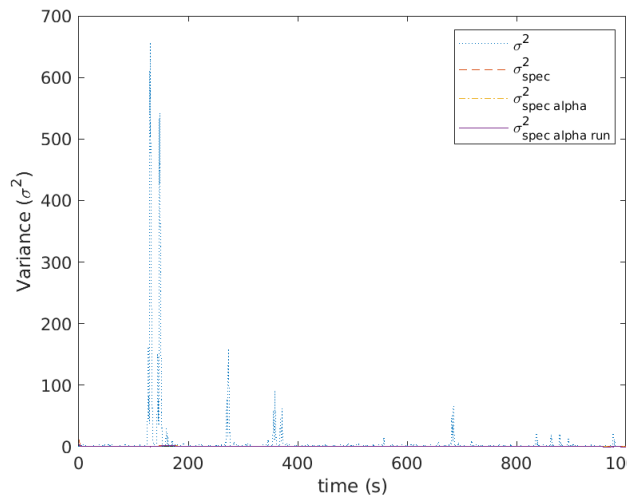


Figure 7.36: Case - C Specific Auxiliary Variances

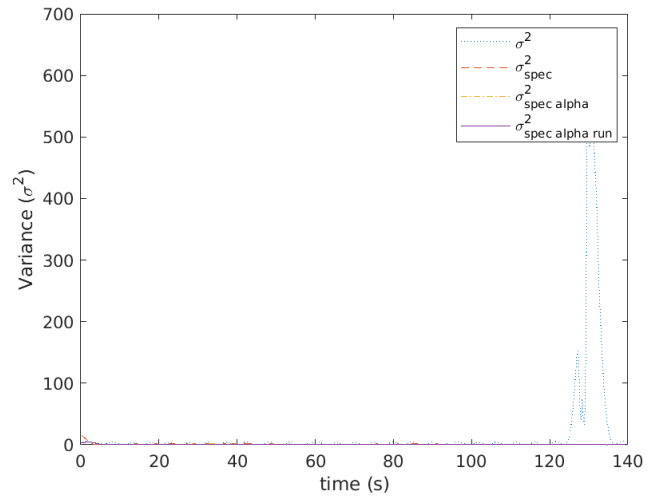


Figure 7.37: Case - C Minimum Time Specific Auxiliary Variances

```

1 %Function to study the initial conditions of the Rossler System for a range
2 %of initial conditions.
3 %Created by Jamell Ivan Samuels.
4 clear all
5 clc
6
7
8 %% Pre-Allocated Values
9
10 Stored_E_bar = [];
11 Stored_E_D_spec_bar = [];
12 Stored_E_D_spec_alpha_bar = [];
13 Stored_E_D_spec_alpha_run_bar = [];
14 Stored_E_D_gen_bar = [];
15 Stored_E_D_gen_alpha_bar = [];
16 Stored_E_D_gen_alpha_run_bar = [];
17 Stored_E_D_sine_bar = [];
18 Stored_E_D_sine_alpha_bar = [];
19 Stored_E_D_sine_alpha_run_bar = [];

```

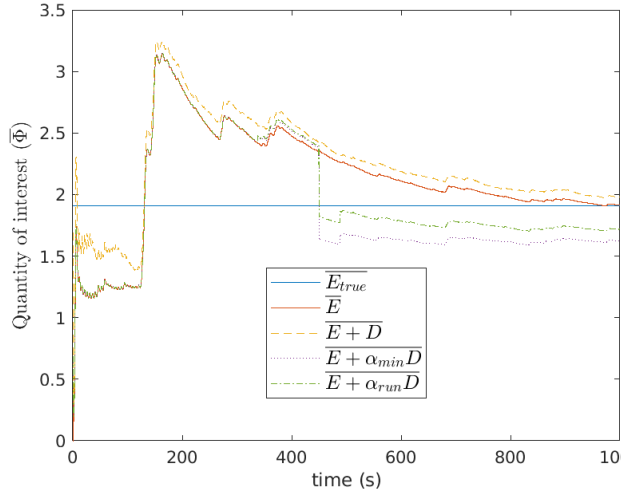


Figure 7.38: Case - C Sine Auxiliary Averages

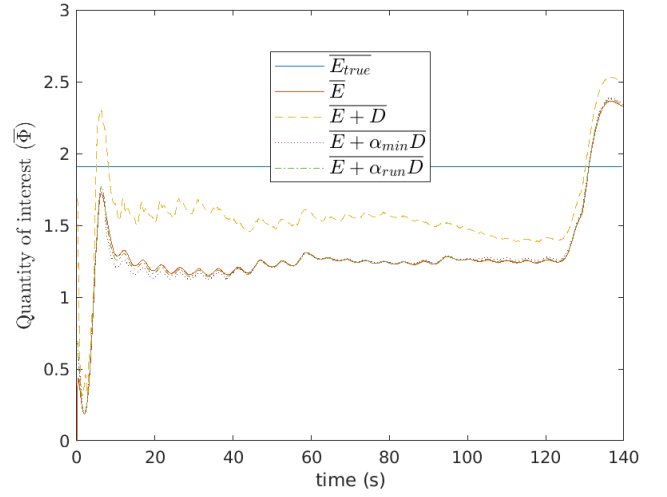


Figure 7.39: Case - C Minimum Time Sine Auxiliary Averages

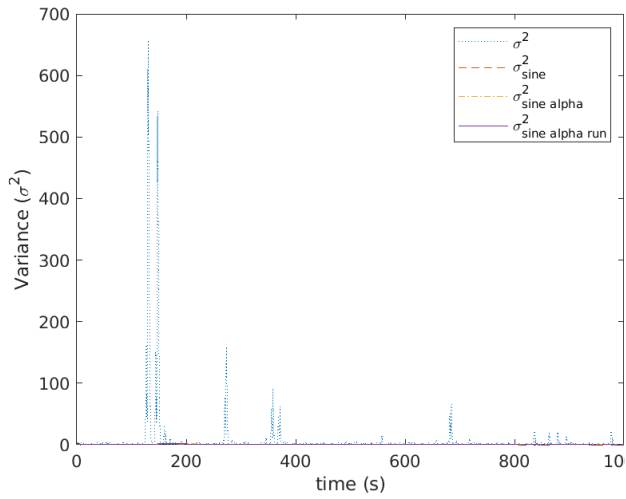


Figure 7.40: Case - C Sine Auxiliary Variances

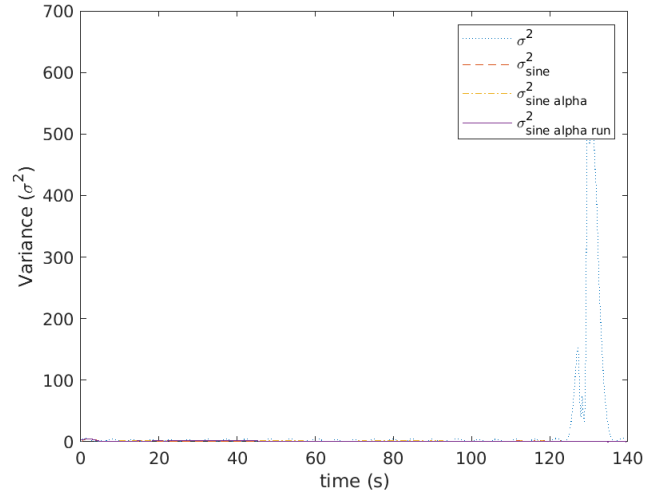


Figure 7.41: Case - C Minimum Time Sine Auxiliary Averages

```

20     Stored_E_std_run_0 = [];
21
22     %MinT Averages
23
24     Stored_MinT_E_bar = [];
25     Stored_MinT_E_D_spec_bar = [];
26     Stored_MinT_E_D_spec_alpha_bar = [];
27     Stored_MinT_E_D_spec_alpha_run_bar = [];
28     Stored_MinT_E_D_gen_bar = [];
29     Stored_MinT_E_D_gen_alpha_bar = [];
30     Stored_MinT_E_D_gen_alpha_run_bar = [];
31     Stored_MinT_E_D_sine_bar = [];
32     Stored_MinT_E_D_sine_alpha_bar = [];
33     Stored_MinT_E_D_sine_alpha_run_bar = [];
34
35     %Variances
36
37     Stored_Var_bar = [];
38     Stored_Var_spec_bar = [];

```



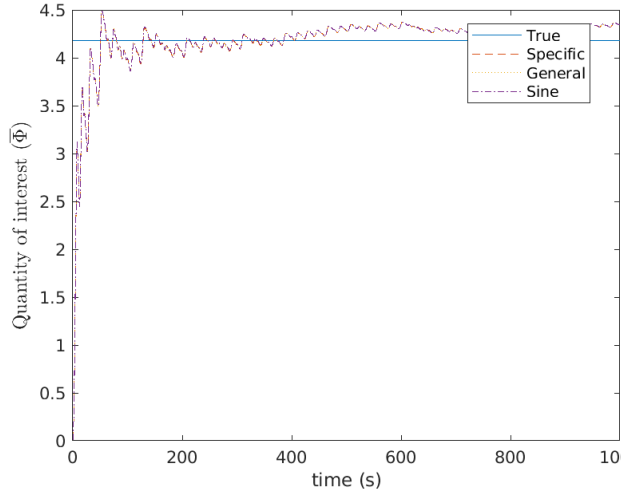


Figure 7.48: Case - E  $\alpha_{run}$  Averages

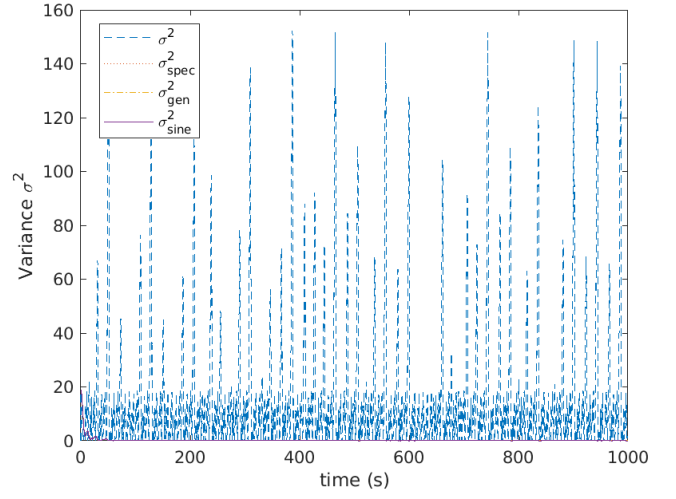


Figure 7.49: Case - E  $\alpha_{run}$  Variances

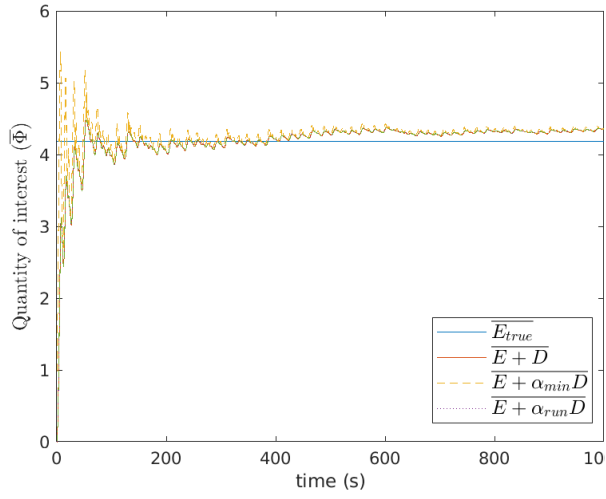


Figure 7.50: Case - E General Auxiliary Averages

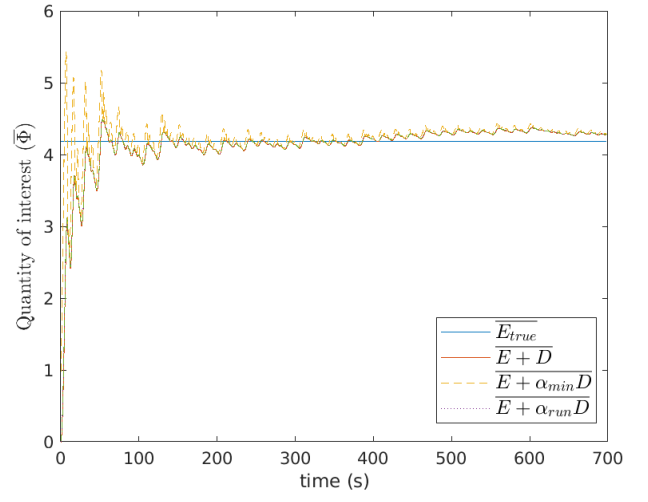


Figure 7.51: Case - E Minimum Time General Auxiliary Averages

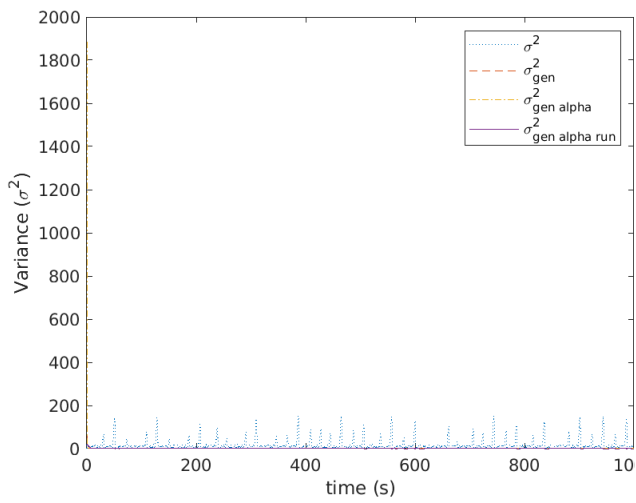


Figure 7.52: Case - E General Auxiliary Variances

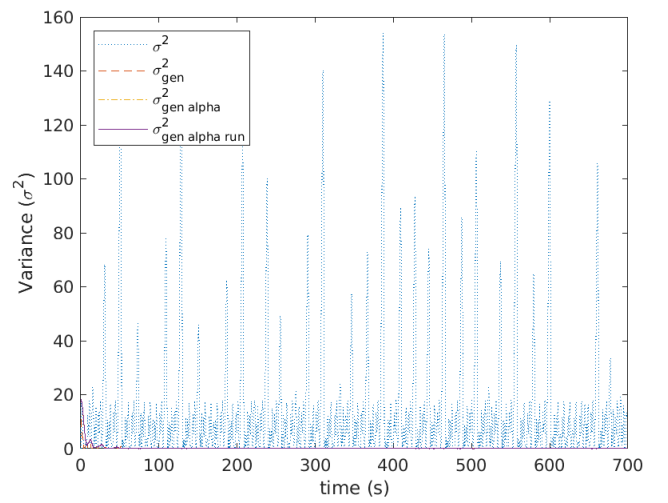


Figure 7.53: Case - E Sine Auxiliary Averages

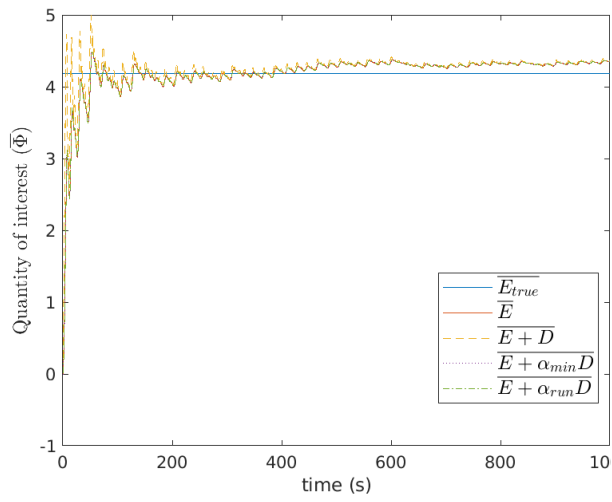


Figure 7.54: Case - E Specific Auxiliary Averages

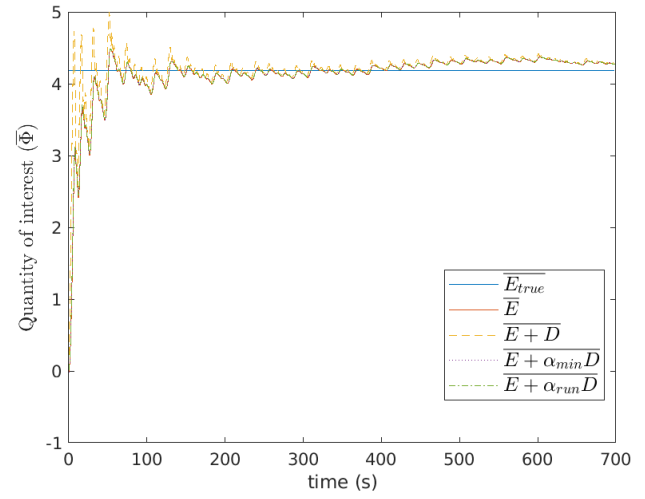


Figure 7.55: Case - E Minimum Time Specific Auxiliary Averages

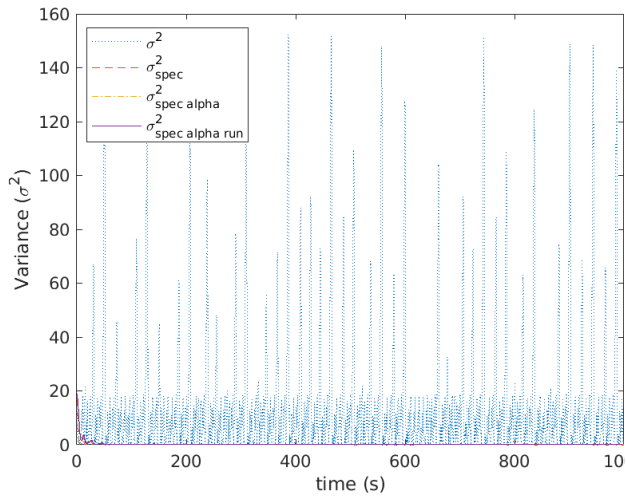


Figure 7.56: Case - E Specific Auxiliary Variances

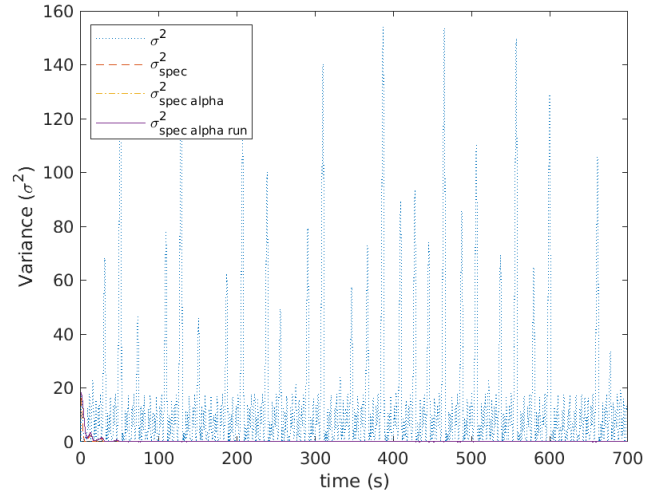


Figure 7.57: Case - E Minimum Time Specific Auxiliary Variances

```

41   Stored_Var_gen_bar = [];
42   Stored_Var_alpha_gen_bar = [];
43   Stored_Var_alpha_run_gen_bar = [];
44   Stored_Var_sine_bar = [];
45   Stored_Var_alpha_sine_bar = [];
46   Stored_Var_alpha_run_sine_bar = [];
47
48   %MinT Variances
49
50   Stored_MinT_Var_bar = [];
51   Stored_MinT_Var_spec_bar = [];
52   Stored_MinT_Var_alpha_spec_bar = [];
53   Stored_MinT_Var_alpha_run_spec_bar = [];
54   Stored_MinT_Var_gen_bar = [];
55   Stored_MinT_Var_alpha_gen_bar = [];
56   Stored_MinT_Var_alpha_run_gen_bar = [];
57   Stored_MinT_Var_sine_bar = [];
58   Stored_MinT_Var_alpha_sine_bar = [];
59   Stored_MinT_Var_alpha_run_sine_bar = [];

```



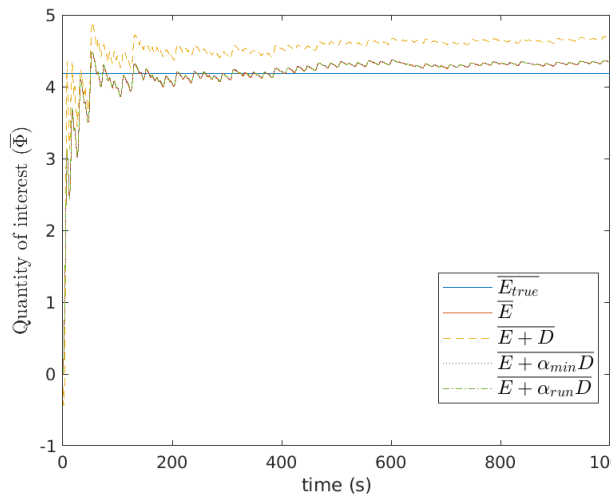


Figure 7.58: Case - E Sine Auxiliary Averages

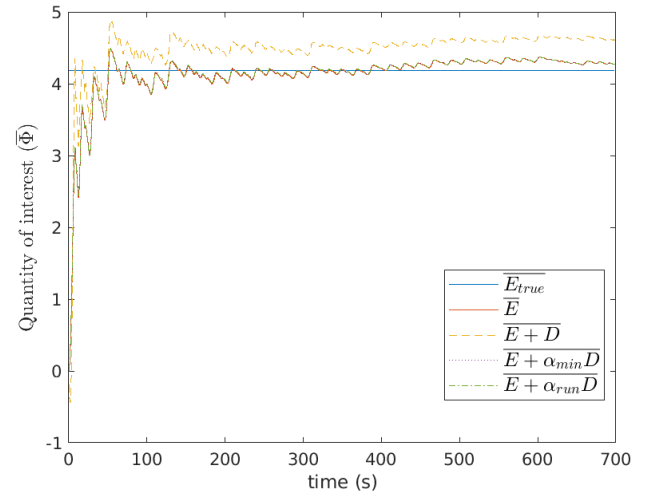


Figure 7.59: Case - E Minimum Time Sine Auxiliary Averages

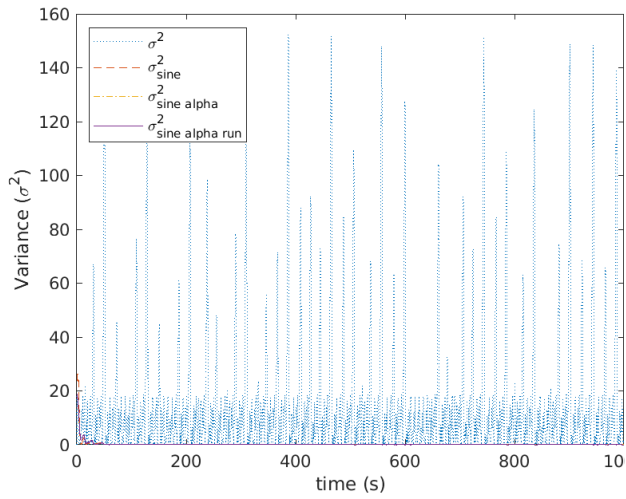


Figure 7.60: Case - E Sine Auxiliary Variances

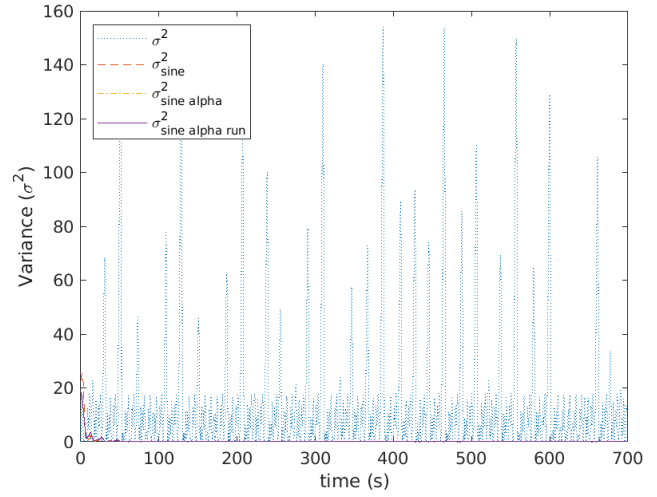


Figure 7.61: Case - E Minimum Time Sine Auxiliary Averages

```

60     Stored_x0 = [];
61
62
63 %% Initial Conditions
64 for p = 1:5;
65
66     clearvars -except Stored_x0 Stored_E_bar Stored_E_D_spec_bar
67         Stored_E_D_spec_alpha_bar Stored_E_D_gen_bar ...
68         Stored_E_D_gen_alpha_bar Stored_E_D_gen_alpha_run_bar Stored_E_D_sine_bar
69         Stored_E_D_sine_alpha_bar ...
70         Stored_E_D_sine_alpha_run_bar Stored_E_std_run_0 Stored_MinT_E_bar
71         Stored_MinT_E_D_spec_bar ...
72         Stored_MinT_E_D_spec_alpha_bar Stored_MinT_E_D_spec_alpha_run_bar
73         Stored_MinT_E_D_gen_bar Stored_MinT_E_D_gen_alpha_bar ...
74         Stored_MinT_E_D_gen_alpha_run_bar Stored_MinT_E_D_sine_bar
75         Stored_MinT_E_D_sine_alpha_bar Stored_MinT_E_D_sine_alpha_run_bar ...
76         Stored_Var_bar Stored_Var_spec_bar Stored_Var_alpha_spec_bar
77         Stored_Var_alpha_run_spec_bar Stored_Var_gen_bar ...
78         Stored_Var_alpha_gen_bar Stored_Var_alpha_run_gen_bar Stored_Var_sine_bar

```

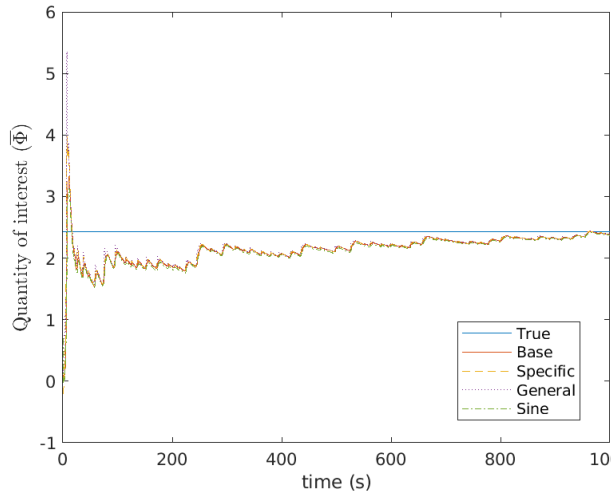


Figure 7.62: Case - F Base Averages

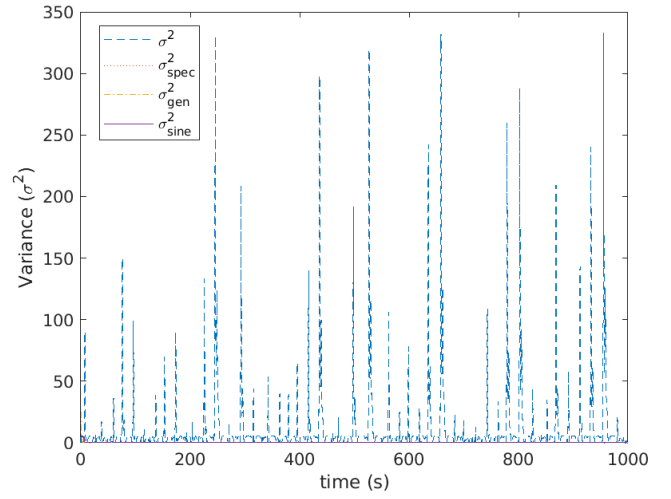


Figure 7.63: Case - F Minimum Time Base Averages

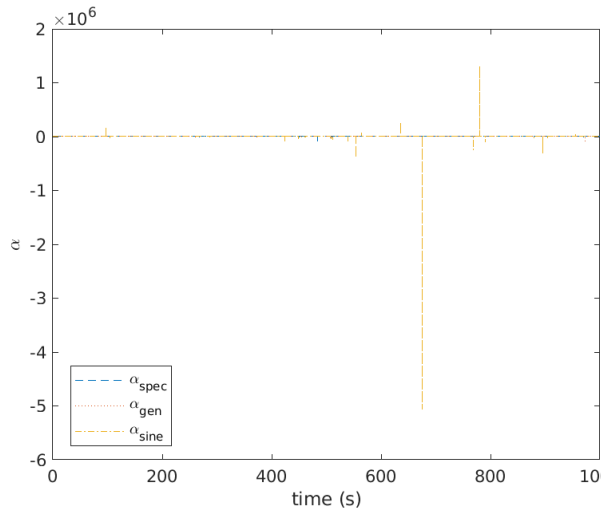


Figure 7.64: Case - F  $\alpha_{min}$

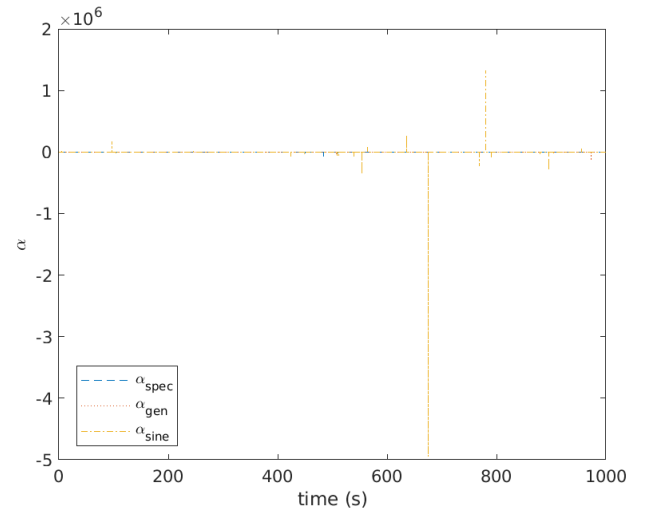


Figure 7.65: Case - F  $\alpha_{run}$

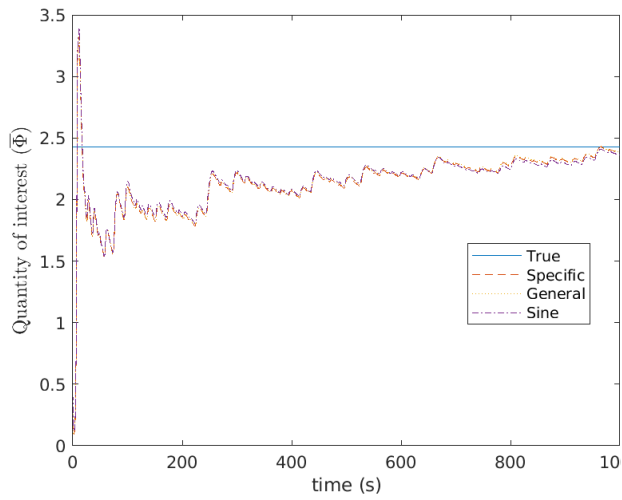


Figure 7.66: Case - F  $\alpha_{min}$  Averages

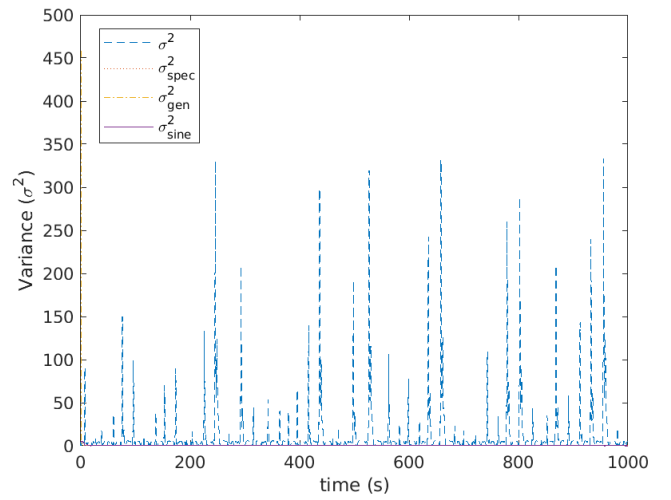


Figure 7.67: Case - F  $\alpha_{min}$  Variances

Stored\_Var\_alpha\_sine\_bar Stored\_Var\_alpha\_run\_sine\_bar ...  
 Stored\_Var\_bar Stored\_Var\_spec\_bar Stored\_Var\_alpha\_spec\_bar

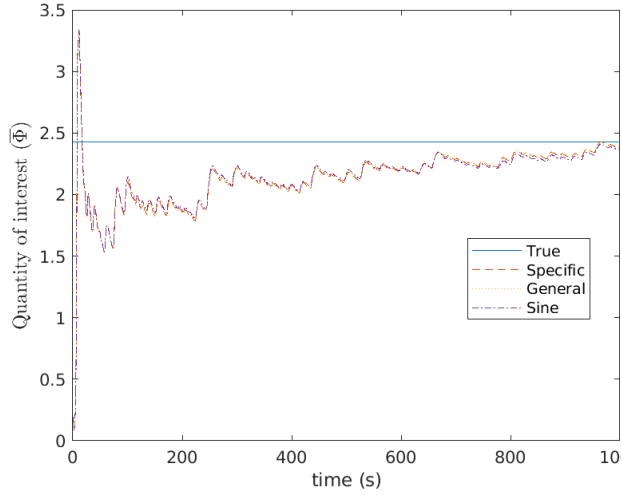


Figure 7.68: Case - F  $\alpha_{run}$  Averages

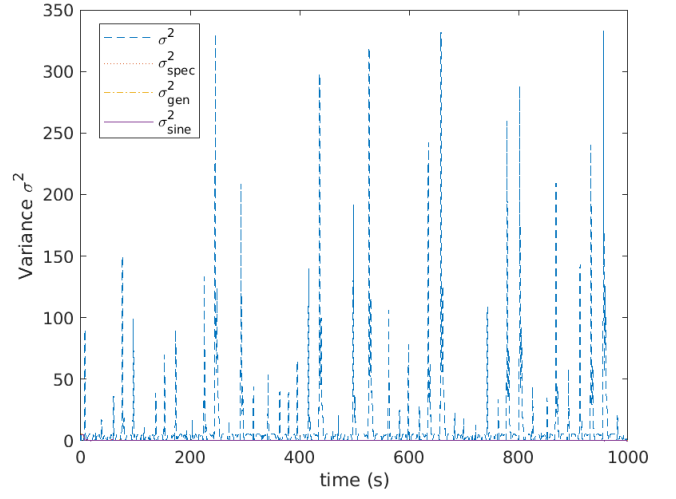


Figure 7.69: Case - F  $\alpha_{run}$  Variances

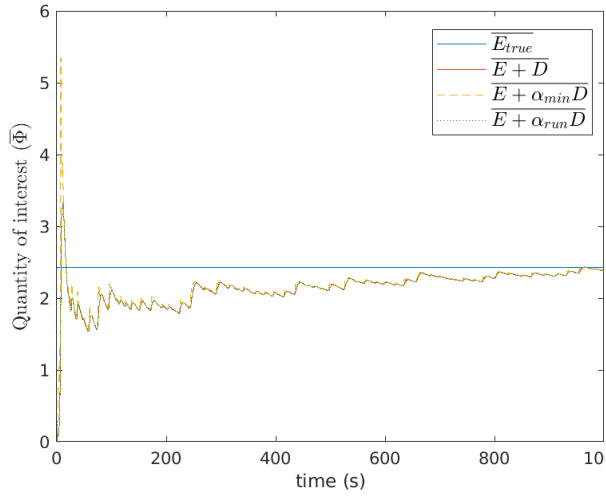


Figure 7.70: Case - F General Auxiliary Averages

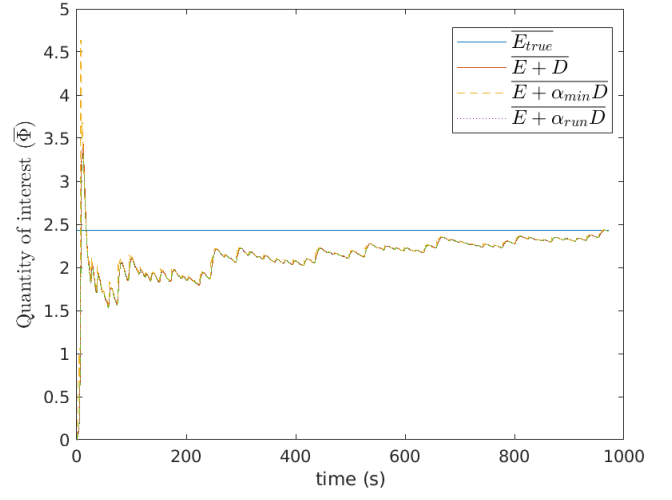


Figure 7.71: Case - F Minimum Time General Auxiliary Averages

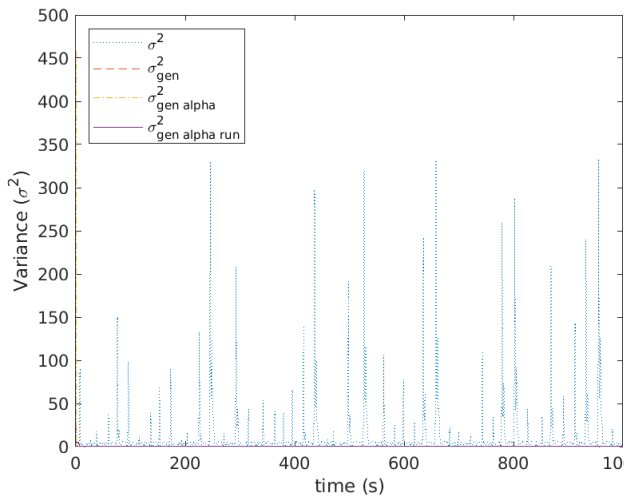


Figure 7.72: Case - F General Auxiliary Variances

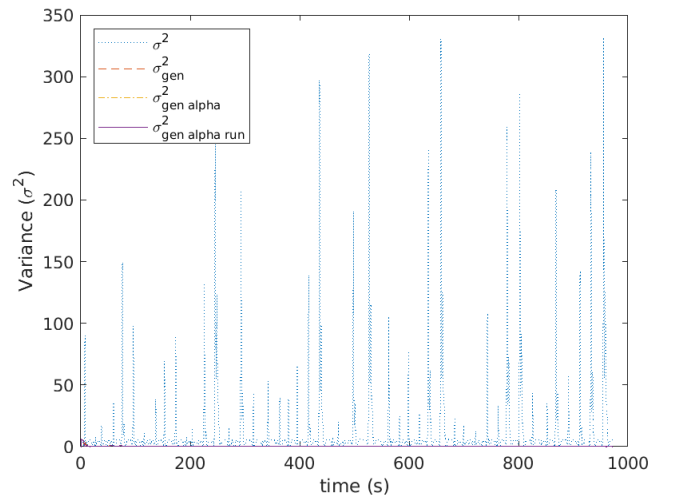


Figure 7.73: Case - F Sine Auxiliary Averages

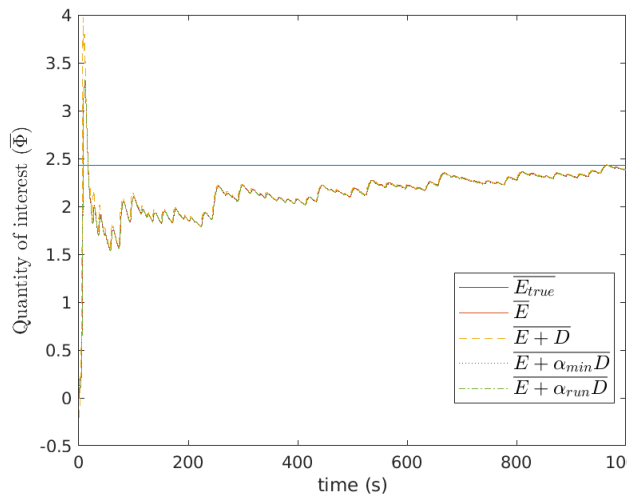


Figure 7.74: Case - F Specific Auxiliary Averages

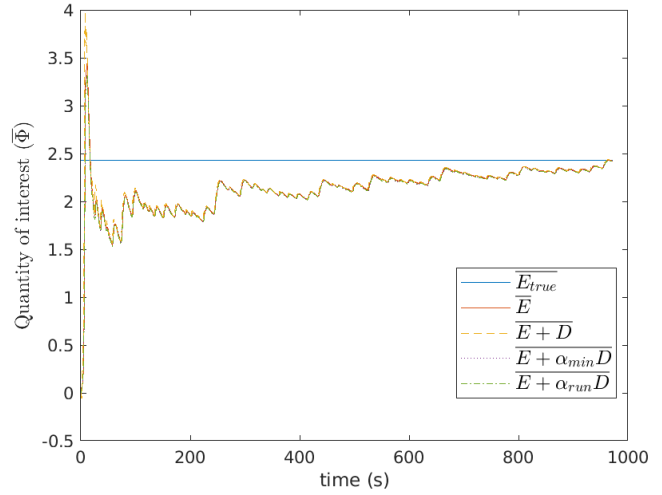


Figure 7.75: Case - F Minimum Time Specific Auxiliary Averages

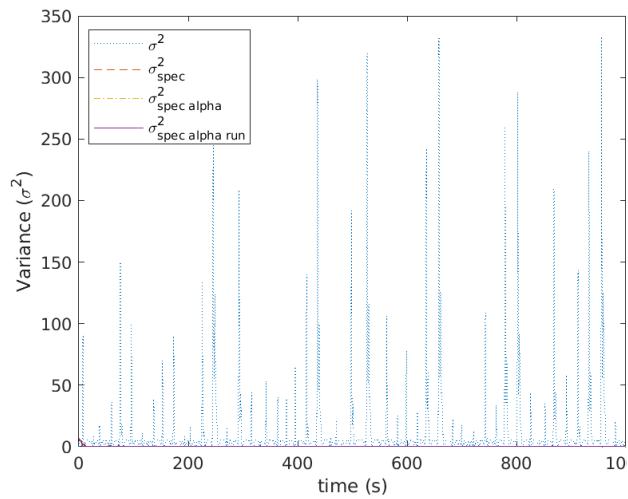


Figure 7.76: Case - F Specific Auxiliary Variances

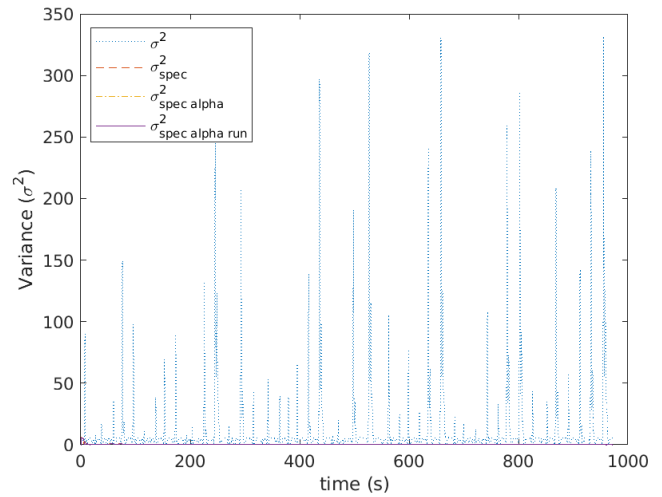


Figure 7.77: Case - F Minimum Time Specific Auxiliary Variances

```

80     Stored_Var_alpha_run_spec_bar Stored_Var_gen_bar ...
81     Stored_Var_alpha_gen_bar Stored_Var_alpha_run_gen_bar Stored_Var_sine_bar
82     Stored_Var_alpha_sine_bar Stored_Var_alpha_run_sine_bar ...
83
84     Stored_MinT_Var_bar Stored_MinT_Var_spec_bar Stored_MinT_Var_alpha_spec_bar
85     Stored_MinT_Var_alpha_run_spec_bar ...
86     Stored_MinT_Var_gen_bar Stored_MinT_Var_alpha_gen_bar
87     Stored_MinT_Var_alpha_run_gen_bar Stored_MinT_Var_sine_bar ...
88     Stored_MinT_Var_alpha_sine_bar Stored_MinT_Var_alpha_run_sine_bar p
89     disp('Current System')
90     disp(p)
91
92     delta_t = 0.1;
93     T = 1000;
94
95     if p == 1; %Case A
96         x0 = [0.014, 0, -0.014];
97         F = @(t, x) [x(2); -x(1)+x(2)*x(3); 1-x(2)^2];
98         tspan = [0:delta_t:T];
99         eps = 0.0000000001;

```

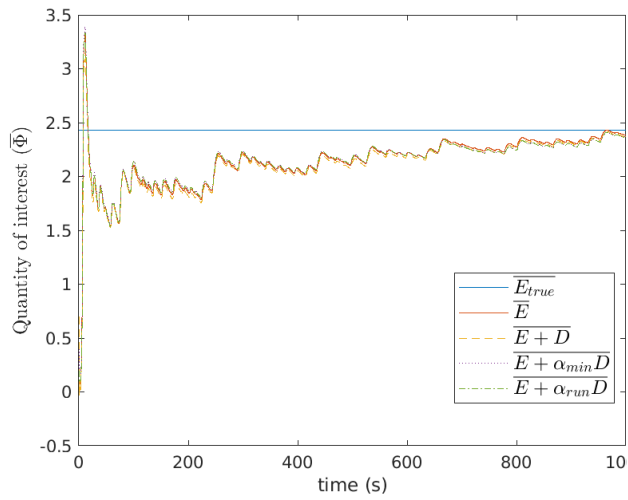


Figure 7.78: Case - F Sine Auxiliary Averages

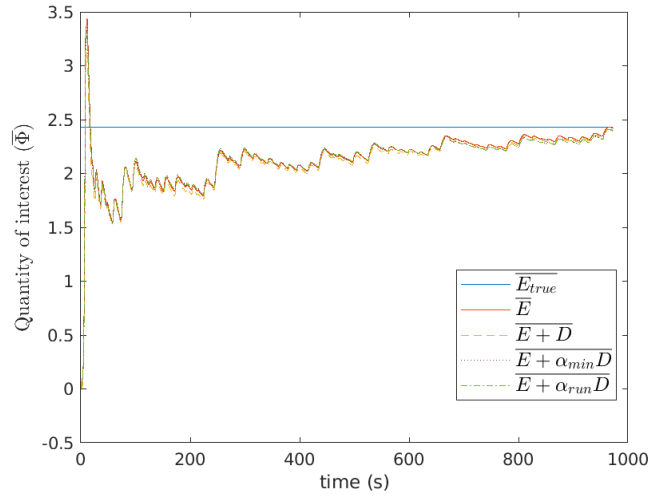


Figure 7.79: Case - F Minimum Time Sine Auxiliary Averages

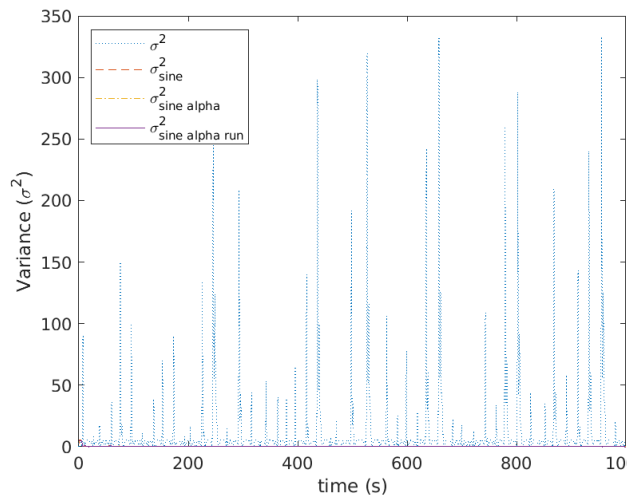


Figure 7.80: Case - F Sine Auxiliary Variances

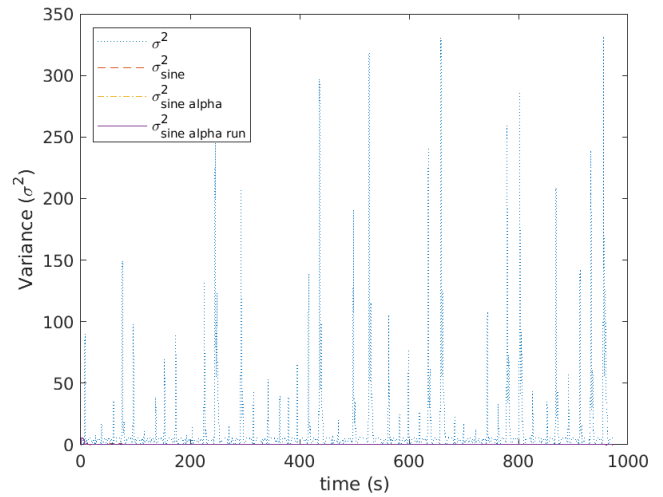


Figure 7.81: Case - F Minimum Time Sine Auxiliary Averages

```

89 options = odeset('RelTol',eps,'AbsTol',[eps eps eps/20]);
90 [t,x] = ode45(F,tspan,x0,options);
91 T = t(end);
92 disp('Time actually ran for')
93 disp(T)
94
95 elseif p == 2; %Case B
96
97 x0 = [0.210,0,-1.20];
98 F = @(t,x) [x(2)*x(3);x(1)-x(2);1-x(1)*x(2)];
99 tspan = [0:delta_t:T];
100 eps = 0.0000000001;
101 options = odeset('RelTol',eps,'AbsTol',[eps eps eps/20]);
102 [t,x] = ode45(F,tspan,x0,options);
103 T = t(end);
104 disp('Time actually ran for')
105 disp(T)
106
107

```

```

108     elseif p == 3; %Case C
109
110     x0=[0.163,0,-1.163];
111     F = @(t,x) [x(2)*x(3);x(1)-x(2);1-x(1)^2];
112     tspan = [0:delta_t:T];
113     eps = 0.0000000001;
114     options = odeset('RelTol',eps,'AbsTol',[eps eps eps/20]);
115     [t,x] = ode45(F,tspan,x0,options);
116     T = t(end);
117     disp('Time actually ran for')
118     disp(T)
119
120     elseif p == 4; %Case F
121
122     x0 = [0.117,0,-0.617];
123     F = @(t,x) [x(2)+x(3);-x(1)+0.5*x(2);x(1)^2-x(3)];
124     tspan = [0:delta_t:T];
125     eps = 0.0000000001;
126     options = odeset('RelTol',eps,'AbsTol',[eps eps eps/20]);
127     [t,x] = ode45(F,tspan,x0,options);
128     T = t(end);
129     disp('Time actually ran for')
130     disp(T)
131
132
133     elseif p == 5; %Case E
134
135     x0=[0.078,0,0.117]
136     F = @(t,x) [x(2)*x(3); x(1)^2-x(2); 1-4*x(1)];
137     tspan = [0:delta_t:T];
138     eps = 0.0000000001;
139     options = odeset('RelTol',eps,'AbsTol',[eps eps eps/20]);
140     [t,x] = ode45(F,tspan,x0,options);
141     T = t(end);
142     disp('Time actually ran for')
143     disp(T)
144     end
145
146
147
148     System_Base_sprott;
149
150
151
152     %Averages
153     Stored_x0(p,:) = x0;
154     Stored_E_bar(p) = E_bar;
155     Stored_E_D_spec_bar(p) = E_D_spec_bar;
156     Stored_E_D_spec_alpha_bar(p) = E_D_spec_alpha_bar;
157     Stored_E_D_spec_alpha_run_bar(p) = E_D_spec_alpha_run_bar;
158     Stored_E_D_gen_bar(p) = E_D_gen_bar;
159     Stored_E_D_gen_alpha_bar(p) = E_D_gen_alpha_bar;
160     Stored_E_D_gen_alpha_run_bar(p) = E_D_gen_alpha_run_bar;
161     Stored_E_D_sine_bar(p) = E_D_sine_bar;
162     Stored_E_D_sine_alpha_bar(p) = E_D_sine_alpha_bar;
163     Stored_E_D_sine_alpha_run_bar(p) = E_D_sine_alpha_run_bar;
164     Stored_E_std_run_0(p) = E_std_run_0;
165
166     %MinT Averages
167

```

```

168 Stored_MinT_E_bar(p) = MinT_E_bar;
169 Stored_MinT_E_D_spec_bar(p) = MinT_E_D_spec_bar;
170 Stored_MinT_E_D_spec_alpha_bar(p) = MinT_E_D_spec_alpha_bar;
171 Stored_MinT_E_D_spec_alpha_run_bar(p) = MinT_E_D_spec_alpha_run_bar;
172 Stored_MinT_E_D_gen_bar(p) = MinT_E_D_gen_bar;
173 Stored_MinT_E_D_gen_alpha_bar(p) = MinT_E_D_gen_alpha_bar;
174 Stored_MinT_E_D_gen_alpha_run_bar(p) = MinT_E_D_gen_alpha_run_bar;
175 Stored_MinT_E_D_sine_bar(p) = MinT_E_D_sine_bar;
176 Stored_MinT_E_D_sine_alpha_bar(p) = MinT_E_D_sine_alpha_bar;
177 Stored_MinT_E_D_sine_alpha_run_bar(p) = MinT_E_D_sine_alpha_run_bar;
178
179 %Variances
180
181 Stored_Var_bar(p) = Var_bar;
182 Stored_Var_spec_bar(p) = Var_spec_bar;
183 Stored_Var_alpha_spec_bar(p) = Var_alpha_spec_bar;
184 Stored_Var_alpha_run_spec_bar(p) = Var_alpha_run_spec_bar;
185 Stored_Var_gen_bar(p) = Var_gen_bar;
186 Stored_Var_alpha_gen_bar(p) = Var_alpha_gen_bar;
187 Stored_Var_alpha_run_gen_bar(p) = Var_alpha_run_gen_bar;
188 Stored_Var_sine_bar(p) = Var_alpha_sine_bar;
189 Stored_Var_alpha_sine_bar(p) = Var_alpha_sine_bar;
190 Stored_Var_alpha_run_sine_bar(p) = Var_alpha_run_sine_bar;
191
192 %MinT Variances
193
194 Stored_MinT_Var_bar(p) = MinT_Var_bar;
195 Stored_MinT_Var_spec_bar(p) = MinT_Var_spec_bar;
196 Stored_MinT_Var_alpha_spec_bar(p) = MinT_Var_alpha_spec_bar;
197 Stored_MinT_Var_alpha_run_spec_bar(p) = MinT_Var_alpha_run_spec_bar;
198 Stored_MinT_Var_gen_bar(p) = MinT_Var_gen_bar;
199 Stored_MinT_Var_alpha_gen_bar(p) = MinT_Var_alpha_gen_bar;
200 Stored_MinT_Var_alpha_run_gen_bar(p) = MinT_Var_alpha_run_gen_bar;
201 Stored_MinT_Var_sine_bar(p) = MinT_Var_sine_bar;
202 Stored_MinT_Var_alpha_sine_bar(p) = MinT_Var_alpha_sine_bar;
203 Stored_MinT_Var_alpha_run_sine_bar(p) = MinT_Var_alpha_run_sine_bar;
204
205 %Variables you should know
206 Stored_T(p) = T;
207 Stored_MinT(p) = minT;
208 Stored_DD_bar_spec(p) = DD_bar_spec;
209 Stored_DD_bar_gen(p) = DD_bar_gen;
210 Stored_DD_bar_sine(p) = DD_bar_sine;
211 Stored_MinT_DD_bar_spec(p) = MinT_DD_bar_spec;
212 Stored_MinT_DD_bar_gen(p) = MinT_DD_bar_gen;
213 Stored_MinT_DD_bar_sine(p) = MinT_DD_bar_sine;
214
215
216
217
218 Stored_DD_bar_spec = nonzeros(Stored_DD_bar_spec);
219 Stored_DD_bar_gen = nonzeros(Stored_DD_bar_gen);
220 Stored_DD_bar_sine = nonzeros(Stored_DD_bar_sine);
221 Stored_MinT_DD_bar_spec = nonzeros(Stored_MinT_DD_bar_spec);
222 Stored_MinT_DD_bar_gen = nonzeros(Stored_MinT_DD_bar_gen);
223 Stored_MinT_DD_bar_sine = nonzeros(Stored_MinT_DD_bar_sine);
224 Stored_T = nonzeros(Stored_T);
225 Stored_MinT = nonzeros(Stored_MinT);
226 Stored_x0 = Stored_x0(any(Stored_x0,2),:); %rows
227 Stored_E_bar = nonzeros(Stored_E_bar);

```

```

228 Stored_E_D_spec_bar = nonzeros(Stored_E_bar);
229 Stored_E_D_spec_alpha_bar = nonzeros(Stored_E_bar);
230 Stored_E_D_gen_bar = nonzeros(Stored_E_D_gen_bar) ;
231 Stored_E_D_gen_alpha_bar = nonzeros(Stored_E_D_gen_alpha_bar);
232 Stored_E_D_gen_alpha_run_bar = nonzeros(Stored_E_D_gen_alpha_run_bar);
233 Stored_E_D_sine_bar = nonzeros(Stored_E_D_sine_bar);
234 Stored_E_D_sine_alpha_bar = nonzeros(Stored_E_D_sine_alpha_bar);
235 Stored_E_D_sine_alpha_run_bar = nonzeros(Stored_E_D_sine_alpha_run_bar);
236 Stored_E_std_run_0 = nonzeros(Stored_E_std_run_0);
237
238
239 %MinT Averages
240
241 Stored_MinT_E_bar = nonzeros(Stored_MinT_E_bar);
242 Stored_MinT_E_D_spec_bar = nonzeros(Stored_MinT_E_D_spec_bar);
243 Stored_MinT_E_D_spec_alpha_bar = nonzeros(Stored_MinT_E_D_spec_alpha_bar);
244 Stored_MinT_E_D_spec_alpha_run_bar = nonzeros(
    Stored_MinT_E_D_spec_alpha_run_bar);
245 Stored_MinT_E_D_gen_bar = nonzeros(Stored_MinT_E_D_gen_bar);
246 Stored_MinT_E_D_gen_alpha_bar = nonzeros(Stored_MinT_E_D_gen_alpha_bar);
247 Stored_MinT_E_D_gen_alpha_run_bar = nonzeros(
    Stored_MinT_E_D_gen_alpha_run_bar);
248 Stored_MinT_E_D_sine_bar = nonzeros(Stored_MinT_E_D_sine_bar);
249 Stored_MinT_E_D_sine_alpha_bar = nonzeros(Stored_MinT_E_D_sine_alpha_bar);
250 Stored_MinT_E_D_sine_alpha_run_bar = nonzeros(
    Stored_MinT_E_D_sine_alpha_run_bar);
251
252 %Variances
253
254 Stored_Var_bar = nonzeros(Stored_Var_bar);
255 Stored_Var_spec_bar = nonzeros(Stored_Var_spec_bar);
256 Stored_Var_alpha_spec_bar = nonzeros(Stored_Var_alpha_spec_bar);
257 Stored_Var_alpha_run_spec_bar = nonzeros(Stored_Var_alpha_run_spec_bar);
258 Stored_Var_gen_bar = nonzeros(Stored_Var_gen_bar);
259 Stored_Var_alpha_gen_bar = nonzeros(Stored_Var_alpha_gen_bar);
260 Stored_Var_alpha_run_gen_bar = nonzeros(Stored_Var_alpha_run_gen_bar);
261 Stored_Var_sine_bar = nonzeros(Stored_Var_sine_bar);
262 Stored_Var_alpha_sine_bar = nonzeros(Stored_Var_alpha_sine_bar);
263 Stored_Var_alpha_run_sine_bar = nonzeros(Stored_Var_alpha_run_sine_bar);
264
265 %MinT Variances
266
267 Stored_MinT_Var_bar = nonzeros(Stored_MinT_Var_bar);
268 Stored_MinT_Var_spec_bar = nonzeros(Stored_MinT_Var_spec_bar);
269 Stored_MinT_Var_alpha_spec_bar = nonzeros(Stored_MinT_Var_alpha_spec_bar);
270 Stored_MinT_Var_alpha_run_spec_bar = nonzeros(
    Stored_MinT_Var_alpha_run_spec_bar);
271 Stored_MinT_Var_gen_bar = nonzeros(Stored_MinT_Var_gen_bar);
272 Stored_MinT_Var_alpha_gen_bar = nonzeros(Stored_MinT_Var_alpha_gen_bar);
273 Stored_MinT_Var_alpha_run_gen_bar = nonzeros(
    Stored_MinT_Var_alpha_run_gen_bar);
274 Stored_MinT_Var_sine_bar = nonzeros(Stored_MinT_Var_sine_bar);
275 Stored_MinT_Var_alpha_sine_bar = nonzeros(Stored_MinT_Var_alpha_sine_bar);
276 Stored_MinT_Var_alpha_run_sine_bar = nonzeros(
    Stored_MinT_Var_alpha_run_sine_bar);
277 end
278
279 %errors
280 E_infty = 2.2
281 E_bar_error = (E_bar-

```



```

282
283
284
285
286
287 plot_function
288
289 MinT_plot_function
290
291 save('All')

```

### 7.3.2 Base Code

```

1  % Rossler System – Base Framework
2  %Created by Jamell Ivan Samuels
3
4
5  %% Initial Conditions
6
7  %delta_t = 0.1;
8  %T = 1000;
9  %x0 = [3.219767,17.917641,-9.154723]; %Case A
10 %a = 0.1;
11 %b = 0.1;
12 %c = 14;
13
14 %% System Solver
15
16 %Differential Equation
17
18 %F = @(t,x) [-x(2)-x(3);x(1)+a*x(2);b+x(3)*(x(1)-c)];
19 %tspan = [0:delta_t:T];
20 %eps = 0.0000000001;
21 %options = odeset('RelTol',eps,'AbsTol',[eps eps eps/20]);
22 %[t,x] = ode45(F,tspan,x0,options);
23 %T = t(end);
24 %disp('Time actually ran for')
25 %disp(T)
26
27 %% Common Variables
28
29 %Calculating E (equation of interest phi)
30
31 E = 0.5 * (power(x(:,1),2) + power(x(:,2),2) + power(x(:,3),2));
32 % Derivatives
33
34 %dxdt = -x(:,2)-x(:,3);
35 %dydt = x(:,1)+a*x(:,2);
36 %dzdt = b+x(:,3).*(x(:,1)-c);
37
38 if p == 1; %Case A
39     dxdt = -x(:,2);
40     dydt = -x(:,1)+ x(:,2).*x(:,3);
41     dzdt = 1 - x(:,2).^2;
42
43 elseif p == 2; %Case B
44     dxdt = x(:,2).*x(:,3);
45     dydt = x(:,1)-x(:,2);
46     dzdt = 1 - x(:,1).*x(:,2);
47
48 elseif p == 3; %Case C

```

```

49     dxdt = x(:,2).*x(:,3);
50     dydt = x(:,1) - x(:,2);
51     dzdt = 1 - x(:,1).^2 ;
52
53     elseif p == 4; %Case F
54         dxdt = x(:,2)+x(:,3);
55         dydt = -x(:,1) +0.5*x(:,2);
56         dzdt = x(:,1).^2-x(:,3);
57
58     elseif p == 5; %Case E
59         dxdt = x(:,2).*x(:,3);
60         dydt = x(:,1).^2 -x(:,2);
61         dzdt = 1-4*x(:,1);
62
63     end
64
65
66
67
68 %% Calculation of Average
69
70 %Step Average of E
71 Steps = round(T/delta_t);
72
73 %Integral of E
74 E_Sum = zeros([Steps,1]);
75 for i=1:Steps
76     E_Sum(i+1) = E_Sum(i)+E(i+1)*delta_t;
77 end
78
79 E_bar = E_Sum(Steps)/T;
80
81 %% Calculation of Running Average
82 E_bar_run = zeros([Steps,1]);
83 for i = 1:Steps
84     E_bar_run(i) = E_Sum(i)/(delta_t*i);
85 end
86
87 %% Calculating Standard Deviation
88 Std = zeros([Steps,1]);
89 for i =1:Steps
90     Std(i) = (E(i)-E_bar);
91 end
92 Std_sum = sum(Std(i));
93
94 Std_bar = Std_sum/sqrt(T);
95
96 %% Calculating Running Standard Deviation
97 Std_run = zeros([Steps,1]);
98 for i = 1:Steps
99     Std_run(i) = (E(i)-E_bar_run(i))/t(i);
100 end
101
102 %% Calculating Variance
103 Var = Std.^2;
104 Var_sum = Std_sum.^2;
105 Var_bar = Var_sum/T;
106
107 %% Calculating Running Variance
108 Var_run = Std_run.^2;

```

```

109
110 %% Specific Auxillary Lyapunov Function
111 Specific_Auxillary;
112
113 %% General Auxillary Lyapunov Function
114 General_Auxillary;
115
116 %% Sine Auxillary Function
117 Sine_Auxillary;
118
119 %% Minum point to Integrate to
120 %METHOD 1
121 %Linear method 1 – based on second order diffenrentials and integrating with
    respect to dt
122 %This method doesn't work as the stationary points are different for x y
123 %and z
124 %k1 = 0;
125 %k2 = 0;
126 %k3 = 0;
127 %x_min = [-1 1]*sqrt(c^2-k3);
128 %y_min = [-1 1].*sqrt(((x_min).^2 - k2)/a^2);
129 %z_min = [-1 1].*sqrt((y_min).^2 - k1);
130
131 %Second Method based on substituting first order differntials into the second
    order differntial
132
133 %for i = 1:Steps
134 %J(i) = b+x(i,3)*(x(i,1)-c)+(x(i,1)+a*x(i,2));
135 %K(i) = (-x(i,2)-x(i,3))+a*(x(i,1)+x(i,2));
136 %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));
137 %end
138
139 %J_min = mink(abs(J),2);
140 %K_min = mink(abs(K),2);
141 %L_min = mink(abs(L),2);
142 %Jmin_loc = find(J==J_min);
143 %Kmin_loc = find(K==K_min);
144 %Lmin_loc = find(L==L_min);
145
146 %Third method based on continous substitution to create an equation to find
147 %an approximate solution.
148 maxE = max(E);
149 minE = min(E);
150 maxE_loc = find(E == maxE);
151 minE_loc = find(E == minE);
152
153 for i = 1:Steps
154     if p == 1; %CaseA
155         d2z(i) = dydt(i) - x(i,1) + x(i,2).*dzdt(i) + x(i,3).*dydt(i) - 2*x(i,3)
            .*dydt(i);
156
157     elseif p == 2; %Case B
158         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);
159
160     elseif p == 3; %Case C
161         d2z(i) = x(i,2).*dzdt(i) + x(i,3).*dydt(i) + dxdt(i) - dydt(i) - 2*x(i
            ,1).*dxdt(i);
162

```

```

163     elseif p ==4; % Case F
164         d2z(i) = dydt(i)+dzdt(i)-x(i,1)+0.5*x(i,2)+2*x(i,1).*dxdt(i)-dzdt(i);
165
166     elseif p == 5; %Case E
167         d2z(i) = x(i,2).*dzdt(i) + dzdt(i).*dydt(i) + 2*dxdt(i).*x(i,1)-dydt(i)
            - 4*dxdt(i);
168     end
169
170 end
171 %d2z(i) = a*(x(i,3)*dxdt(i))-x(i,1)*dzdt(i)-c*dzdt(i); for rossler
172
173 d2z_min = mink(abs(d2z),8);
174
175 d2zmin_loc(:,1) = find(abs(d2z) == d2z_min(1));
176 d2zmin_loc(:,2) = find(abs(d2z) == d2z_min(2));
177 d2zmin_loc(:,3) = find(abs(d2z) == d2z_min(3));
178 d2zmin_loc(:,4) = find(abs(d2z) == d2z_min(4));
179 d2zmin_loc(:,5) = find(abs(d2z) == d2z_min(5));
180 d2zmin_loc(:,6) = find(abs(d2z) == d2z_min(6));
181 %d2zmin_loc(:,7) = find(abs(d2z) == d2z_min(7));
182 %d2zmin_loc(:,8) = find(abs(d2z) == d2z_min(8));
183
184 min_d2z = find(d2zmin_loc < minE_loc);
185 if isempty(min_d2z) == 1
186     min_d2z = minE_loc
187     disp('minimum E value used for first stationairy point, not good')
188 end
189 min_d2z = max(min_d2z);
190 max_d2z = d2zmin_loc(d2zmin_loc > maxE_loc & d2zmin_loc > min_d2z);
191 max_d2z = min(max_d2z);
192
193 if isempty(max_d2z) == 1
194     max_d2z = maxE_loc
195     disp('maximum E value used for second stationary point, not good')
196 end
197
198
199 minT(:,1) = min_d2z;
200 minT(:,2) = max_d2z;
201
202
203
204 minT = sort(minT);
205 %Note that from this point onwards the value of Steps has Changed
206 Steps = minT(2)-minT(1); %new number of Steps
207 ind1 = minT(1);
208 ind2 = minT(2);
209 minT = Steps*delta_t; % reassigning Minimum T to the minimum time
210 t_m = t(ind1:ind2);
211 disp('The minimum time is')
212 disp(minT)
213
214 %% Calculation of Minimum Time Average
215
216 MinT_E = E(ind1:ind2);
217 %Integral of E
218 MinT_E_Sum = zeros([Steps,1]);
219 for i=1:Steps
220     MinT_E_Sum(i+1) = MinT_E_Sum(i)+MinT_E(i)*delta_t;
221 end

```

```

222
223 MinT_E_bar = MinT_E.Sum(Steps)/(delta_t*Steps);
224
225 %% Calculation of Minimum Time Running Average
226 MinT_E_bar_run = zeros([Steps,1]);
227 for i = 1:Steps
228     MinT_E_bar_run(i) = MinT_E.Sum(i)/(delta_t*i);
229 end
230
231 %% Calculating Variance - Specific
232 MinT_Var = zeros([Steps,1]);
233 for i = 1:Steps
234     MinT_Var(i) = (MinT_E(i)-MinT_E_bar)^2;
235 end
236 MinT_Var_sum = sum(MinT_Var(i));
237 MinT_Var_bar = MinT_Var_sum/(delta_t*Steps);
238
239 %% Calculating Running Variance - Specific
240 MinT_Var_run = zeros([Steps,1]);
241 for i = 1:Steps
242     MinT_Var_run(i) = (MinT_E(i)-MinT_E_bar_run(i))^2;
243 end
244
245 %% Specific Auxillary Lyapunov Minimum Time
246 Specific_Auxillary_MinT;
247
248 %% General Auxillary Lypunov Minimum Time
249 General_Auxillary_minT;
250
251 %% Sine Auxillary Minimum Time
252 Sine_Auxillary_MinT;
253
254 %% Sum of Sigma = 0
255 min_std_run = min(abs(Std_run));
256 min_std_loc = find(abs(Std_run) == min_std_run);
257 E_std_run_0 = E_bar_run(min_std_loc);
258
259
260
261
262
263
264
265 %save('System_Base')

```

### 7.3.3 Specific Auxiliary

```

1 %Function to calculate the Specific Auxillary Case
2 %Created by Jamell Ivan Samuels
3
4 %% Finding the Specific Auxillary Function
5 %Calucldaing Lyapunov
6 %syms x1 x2 x3;
7 %xdot1 = -x2-x3;
8 %xdot2 = x1+a*x2;
9 %xdot3 = b+x3*(x1-c);
10 syms V(x1,x2, x3)
11 V(x1,x2,x3) = x1^2 + x1*x2 + x2^2 +x2*x3 +x1*x3 +x3^2
12 disp('Lypunov Generated')
13 %% Forming the Specific Lyapunov Auxillary Function
14 D1_spec = diff(V,x1); %(V_1=x)

```

```

15 D2_spec = diff(V,x2); %(V_2=y)
16 D3_spec = diff(V,x3);%(V_3=z)
17
18 x1 = x(:,1);
19 x2 = x(:,2);
20 x3 = x(:,3);
21 DD_spec(:,1) = subs(D1_spec);
22 DD_spec(:,2) = subs(D2_spec);
23 DD_spec(:,3) = subs(D3_spec);
24 DD_spec(:,1) = DD_spec(:,1).*dxdt;
25 DD_spec(:,2) = DD_spec(:,2).*dydt;
26 DD_spec(:,3) = DD_spec(:,3).*dzdt;
27 disp('DD matrices calculated')
28 %% Calculating Averages of DD
29 %% Average of DD
30 %Step Average of DD
31
32 %Integral of E + D
33 %Summation of D
34 DDSum_spec = zeros([Steps,1]);
35 for i = 1:Steps
36 DDSum_spec(i) = DD_spec(i,1)+DD_spec(i,2)+DD_spec(i,3);
37 end
38 DD_bar_spec = sum(DDSum_spec)/T;
39 %Calculating D^2 Bar
40 DD_spec_squared= zeros([Steps,1]);
41 for i = 1:Steps
42     DD_spec_squared(i) = (DDSum_spec(i)).^2;
43 end
44 DD_spec_squared_Sum = sum(DD_spec_squared);
45 DD_spec_squared_bar = DD_spec_squared_Sum/T; %DD Squared bar
46 %% Average of E & DD
47 %Combining of E & DD
48 Comb_E_D_spec = zeros([Steps,1]);
49 for i = 1:Steps
50 Comb_E_D_spec(i) = E(i)+DDSum_spec(i);
51 end
52
53 %Summation of E & DD
54 Sum_E_D_spec = zeros([Steps,1]);
55 for i = 1:Steps
56     Sum_E_D_spec(i+1) = (Sum_E_D_spec(i))+(Comb_E_D_spec(i)*delta_t);
57 end
58
59 %Average of E + D
60
61 E_D_spec_bar = sum(Sum_E_D_spec(Steps))/T;
62
63 %% The Changing Average
64
65 for i = 1:Steps
66 E_D_spec_t(i) = sum(Sum_E_D_spec(i))/t(i); %Average of E + D specific
67 end
68
69
70 E_D_spec_t = transpose(E_D_spec_t);
71
72
73
74 %% Variance of Specific Auxillary

```

```

75  for i = 1:Steps
76  Var_spec(i) = (E_D_spec_t(i) - E_D_spec_bar).^2;
77  end
78
79  Var_spec_bar = sum(Var_spec(Steps))/T;
80
81
82  %% Calculation of alpha - Specific
83
84  for i = 1:Steps
85      disp(i)
86      alpha_spec(i) = (E(i)-E_bar)/(DD_spec_squared(i));
87  end
88
89  alpha_spec = alpha_spec/T;
90
91  alpha_spec = transpose(alpha_spec);
92
93  %alpha_test(1) =
94
95  %% Calculating Average of (E + alphaD) - Specific
96
97  %Combining E + alpha_bar D
98
99  E_D_spec_alpha = zeros([Steps,1]);
100  for i = 1:Steps
101  E_D_spec_alpha(i+1) = E_D_spec_alpha(i) + (E(i)+alpha_spec(i).*DDSum_spec(i))*
      delta_t;
102  end
103
104  %Averaging E +ialphaD
105
106
107  Sum_E_D_spec_alpha = sum(E_D_spec_alpha(Steps));
108  E_D_spec_alpha_bar = Sum_E_D_spec_alpha/T;
109
110  % Changing AVerage E +alpha D
111  for i = 1:Steps
112      E_D_spec_alpha_t(i) = sum(E_D_spec_alpha(i))/t(i);
113  end
114
115  E_D_spec_alpha_t = transpose(E_D_spec_alpha_t);
116
117  %% Standard Deviation of E+alpha D
118
119  Std_alpha_spec = zeros([Steps,1]);
120  for i =1:Steps
121      Std_alpha_spec(i) = (E_D_spec_alpha_t(i)-E_D_spec_alpha_bar);
122  end
123  Std_alpha_spec_sum = sum(Std_alpha_spec(Steps));
124  Std_alpha_spec_bar = Std_alpha_spec_sum/sqrt(T);
125
126
127  %% Variance of E+alpha D
128
129  Var_alpha_spec = Std_alpha_spec.^2;
130  Var_alpha_spec_sum = Std_alpha_spec_sum.^2
131  Var_alpha_spec_bar = Var_alpha_spec_sum/T;
132
133

```

```

134 %% Calculating Running Alpha
135
136 for i = 1:Steps
137     disp(i)
138     alpha_spec_run(i) = (E(i)-E_bar_run(i))/(DD_spec_squared(i));
139 end
140
141 alpha_spec_run = alpha_spec_run/T;
142
143 alpha_spec_run = transpose(alpha_spec_run);
144
145 %% Calculating E +alpha D run
146
147 E_D_spec_alpha_run = zeros([Steps,1]);
148 for i = 1:Steps
149     E_D_spec_alpha_run(i+1) = E_D_spec_alpha_run(i)+(E(i)+alpha_spec_run(i).*(
        DDSum_spec(i)))*delta_t;
150 end
151
152 %Averaging E +ialphaD
153
154
155
156 Sum_E_D_spec_alpha_run = sum(E_D_spec_alpha_run(Steps));
157 E_D_spec_alpha_run_bar = Sum_E_D_spec_alpha_run/T;
158
159 for i = 1:Steps
160     E_D_spec_alpha_run_t(i) = sum(E_D_spec_alpha_run(i))/t(i);
161 end
162
163 E_D_spec_alpha_run_t = transpose(E_D_spec_alpha_run_t);
164
165
166
167
168 %% Standard Deviation of E+alpha D run
169
170 Std_alpha_run_spec = zeros([Steps,1]);
171 for i = 1:Steps
172     Std_alpha_run_spec(i) = (E_D_spec_alpha_run_t(i)-E_D_spec_alpha_run_bar);
173 end
174 Std_alpha_run_spec_sum = sum(Std_alpha_run_spec(Steps));
175 Std_alpha_run_spec_bar = Std_alpha_run_spec_sum/sqrt(T);
176
177
178 %% Variance of E+alpha D run
179 Var_alpha_run_spec = Std_alpha_run_spec.^2;
180 Var_alpha_run_spec_sum = Std_alpha_run_spec_sum.^2;
181 Var_alpha_run_spec_bar = Var_alpha_run_spec_sum/T;

```

### 7.3.4 General Auxiliary

```

1 %Function to calculate the Genera Auxillary Minimum Time Lyapunov Function
2
3 %Created by Jamell Ivan Samuels
4
5
6 %% Forming the General Auxillary Lyapunov Function
7
8
9 x1 = x(:,1);

```



```

10 x2 = x(:,2);
11 x3 = x(:,3);
12
13
14 DD_gen(:,1) = dxdt;
15 DD_gen(:,2) = dydt;
16 DD_gen(:,3) = dzdt;
17 DD_gen(:,4) = x(:,1).*dydt + x(:,2).*dxdt;
18 DD_gen(:,5) = x(:,1).*dzdt + x(:,3).*dxdt;
19 DD_gen(:,6) = x(:,3).*dydt + x(:,2).*dzdt;
20 DD_gen(:,7) = x(:,1).*x(:,3).*dydt + x(:,1).*x(:,2).*dzdt + x(:,2).*x(:,3).*dxdt;
21 DD_gen(:,8) = 2.*x(:,1).*dxdt;
22 DD_gen(:,9) = 2.*x(:,2).*dydt;
23 DD_gen(:,10) = 2.*x(:,3).*dzdt;
24 %% Calculating Averages of DD
25 %% Average of DD
26 %Step Average of DD
27
28 %Integral of E + D
29 %Summation of D
30 DDSum_gen = zeros([Steps,1]);
31 for i = 1:Steps
32 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);
33 end
34 DD_bar_gen = sum(DDSum_gen)/T;
35 %Calculating D^2 Bar
36 DD_gen_squared= zeros([Steps,1]);
37 for i = 1:Steps
38     DD_gen_squared(i) = (DDSum_gen(i)).^2;
39 end
40 DD_gen_squared_Sum = sum(DD_gen_squared);
41 DD_gen_squared_bar = DD_gen_squared_Sum/T; %DD Squared bar
42 %% Average of E & DD
43 %Combining of E & DD
44 Comb_E_D_gen = zeros([Steps,1]);
45 for i = 1:Steps
46 Comb_E_D_gen(i) = E(i)+DDSum_gen(i);
47 end
48
49 %Summation of E & DD
50 Sum_E_D_gen = zeros([Steps,1]);
51 for i = 1:Steps
52     Sum_E_D_gen(i+1) = (Sum_E_D_gen(i))+(Comb_E_D_gen(i)*delta_t);
53 end
54
55 %Average of E + D
56
57 for i = 1:Steps
58     E_D_gen_t(i) = sum(Sum_E_D_gen(i))/t(i);
59 end
60 E_D_gen_t = transpose(E_D_gen_t);
61
62 E_D_gen_bar = sum(Sum_E_D_gen(Steps))/T; %Average of E + D specific
63
64 %% Variance of General Auxillary
65
66 for i = 1:Steps
67     Var_gen(i) = ((E_D_gen_t(i)-E_D_gen_bar).^2)/(delta_t*i);
68 end

```

```

69
70 Var_gen_bar = sum(Var_gen(Steps))/T;
71
72
73 %% Calculation of alpha min – General
74 for i = 1:Steps
75     disp(i)
76     alpha_gen(i) = (E(i)-E_bar)/(DD_gen_squared(i));
77 end
78
79 alpha_gen = alpha_gen/T;
80
81 alpha_gen = transpose(alpha_gen);
82
83 %% Calculating Average of (E + alphaD) – General
84
85 %Combining E + alpha_bar D
86
87 E_D_gen_alpha = zeros([Steps,1]);
88 for i = 1:Steps
89     E_D_gen_alpha(i+1) = E_D_gen_alpha(i)+(E(i)+alpha_gen(i).*(DDSum_gen(i)))*
        delta_t;
90 end
91
92 %Averaging E +ialphaD
93 for i = 1:Steps
94     E_D_gen_alpha_t(i) = sum(E_D_gen_alpha(i))/t(i);
95 end
96 E_D_gen_alpha_t = transpose(E_D_gen_alpha_t);
97
98
99 Sum_E_D_gen_alpha = sum(E_D_gen_alpha(Steps));
100 E_D_gen_alpha_bar = Sum_E_D_gen_alpha/T;
101
102
103
104 %% Standard Deviation of E+alpha D
105
106 Std_alpha_gen = zeros([Steps,1]);
107 for i = 1:Steps
108     Std_alpha_gen(i) = (E_D_gen_alpha_t(i)-E_D_gen_alpha_bar)/(t(i));
109 end
110 Std_alpha_gen_sum = sum(Std_alpha_gen(Steps));
111 Std_alpha_gen_bar = Std_alpha_gen_sum/sqrt(T);
112
113 %% Variance of E +alpha D
114 Var_alpha_gen = Std_alpha_gen.^2;
115 Var_alpha_gen_sum = Std_alpha_gen_sum.^2;
116 Var_alpha_gen_bar = Var_alpha_gen_sum/T;
117
118 %% Calculating Running Alpha – General
119
120 for i = 1:Steps
121     disp(i)
122     alpha_gen_run(i) = (E(i)-E_bar_run(i))/(DD_gen_squared(i));
123 end
124
125 alpha_gen_run = alpha_gen_run/T;
126
127 alpha_gen_run = transpose(alpha_gen_run);

```

```

128
129 %% Calculating E +alpha D run - General
130
131 E_D_gen_alpha_run = zeros([Steps,1]);
132 for i = 1:Steps
133     E_D_gen_alpha_run(i+1) = E_D_gen_alpha_run(i) + (E(i)+alpha_gen_run(i).*(
        DDSum_gen(i)))*delta_t;
134 end
135
136 %Averaging E +ialphaD
137
138 for i = 1:Steps
139     E_D_gen_alpha_run_t(i) = sum(E_D_gen_alpha_run(i))/t(i);
140 end
141
142
143 E_D_gen_alpha_run_t = transpose(E_D_gen_alpha_run_t);
144
145 Sum_E_D_gen_alpha_run = sum(E_D_gen_alpha_run(Steps));
146 E_D_gen_alpha_run_bar = Sum_E_D_gen_alpha_run/T;
147
148
149 %% Standard Deviation of E+alpha D run - General
150
151 Std_alpha_run_gen = zeros([Steps,1]);
152 for i = 1:Steps
153     Std_alpha_run_gen(i) = (E_D_gen_alpha_run_t(i)-E_D_gen_alpha_run_bar);
154 end
155 Std_alpha_run_gen_sum = sum(Std_alpha_run_gen(Steps));
156 Std_alpha_run_gen_bar = Std_alpha_run_gen_sum/sqrt(T);
157
158
159 %% Variance of E+alpha D run - General
160 Var_alpha_run_gen = Std_alpha_run_gen.^2;
161 Var_alpha_run_gen_sum = Std_alpha_run_gen_sum.^2;
162 Var_alpha_run_gen_bar = Var_alpha_run_gen_sum/T;
163
164
165 %%

```

### 7.3.5 Sine Auxiliary

```

1
2 %Function to calculate the Sine Auxillary Case
3 %Created by Jamell Ivan Samuels
4
5 %% Finding the Specific Auxillary Function
6
7 %% Forming the Sine Auxillary Function
8
9 E1 = E(1);
10 E2 = E(Steps);
11 for i = 1:Steps
12     dEdt(i) = x(i,1)+x(i,2)+x(i,3)
13 end
14 for i = 1:Steps
15     DD_sine(i) = -dEdt(i).*cos((E1-E(i))/(E1-E2)*pi);
16 end
17 DD_sine = transpose(DD_sine)
18
19

```

```

20 %% Calculating Averages of DD
21 %% Average of DD
22 %Step Average of DD
23
24 %Integral of E + D
25 %Summation of D
26
27 DD_bar_sine = sum(DD_sine)/T;
28 %Calculating D^2 Bar
29 for i = 1:Steps
30 DD_sine_squared(i) = DD_sine(i).^2 ;
31 end
32 DD_sine_squared = transpose(DD_sine_squared);
33 DD_sine_squared_Sum = sum(DD_sine_squared);
34 DD_sine_squared_bar = DD_sine_squared_Sum/T; %DD Squared bar
35 %% Average of E & DD
36 %Combining of E & DD
37 Comb_E_D_sine = zeros([Steps,1]);
38 for i = 1:Steps
39 Comb_E_D_sine(i) = E(i)+DD_sine(i);
40 end
41
42 %Summation of E & DD
43 Sum_E_D_sine = zeros([Steps,1]);
44 for i = 1:Steps
45 Sum_E_D_sine(i+1) = Sum_E_D_sine(i)+Comb_E_D_sine(i)*delta_t;
46 end
47
48 %Average of E + D
49
50 for i = 1:Steps
51 E_D_sine_t(i) = sum(Sum_E_D_sine(i))/t(i);
52 end
53
54 E_D_sine_t = transpose(E_D_sine_t);
55
56 E_D_sine_bar = sum(Sum_E_D_sine(Steps))/T; %Average of E + D specific
57
58 %% Variance of Sine Auxillary
59
60 for i = 1:Steps
61 Var_sine(i) = (E_D_sine_t(i) - E_D_sine_bar).^2;
62 end
63
64 Var_sine_bar = sum(Var_sine(Steps))/T;
65
66 %% Calculation of alpha min - Sine
67
68 for i = 1:Steps
69 disp(i)
70 alpha_sine(i) = (E(i)-E_bar)/(DD_sine_squared(i));
71 end
72
73 alpha_sine = alpha_sine/T;
74
75 alpha_sine = transpose(alpha_sine);
76 %alpha_sine(1) = 0; % THis actually doesn't make
77 %alpha_sine(end) = 0;
78
79

```

```

80 %% Calculating Average of (E + alphaD) - Sine
81
82 %Combining E + alpha_bar D
83
84 E_D_sine_alpha = zeros([Steps,1]);
85 for i = 1:Steps
86 E_D_sine_alpha(i+1) = E_D_sine_alpha(i)+(E(i)+alpha_sine(i).*DD_sine(i))*delta_t
87 ;
88 end
89
90 %Averaging E +ialphaD
91
92 for i = 1:Steps
93 E_D_sine_alpha_t(i) = sum(E_D_sine_alpha(i))/t(i);
94 end
95 E_D_sine_alpha_t = transpose(E_D_sine_alpha_t);
96
97 Sum_E_D_sine_alpha = sum(E_D_sine_alpha(Steps));
98 E_D_sine_alpha_bar = Sum_E_D_sine_alpha/T;
99
100 %% Standard Deviation of E +alpha D
101
102 Std_alpha_sine = zeros([Steps,1]);
103 for i = 1:Steps
104 Std_alpha_sine(i) = (E_D_sine_alpha_t(i)-E_D_sine_alpha_bar); % This is is
105 the sample deviation
106 end
107 Std_alpha_sine_sum = sum(Std_alpha_sine(Steps));
108 Std_alpha_min_sine_bar = Std_alpha_sine_sum/sqrt(T); %This is the standard
109 deviation
110
111 %% Variance of E+alpha D
112 Var_alpha_sine = Std_alpha_sine.^2;
113 Var_alpha_sine_sum = Std_alpha_sine_sum.^2;
114 Var_alpha_sine_bar = Var_alpha_sine_sum/T;
115
116 %% Calculating Running Alpha
117
118 for i = 1:Steps
119 disp(i)
120 alpha_sine_run(i) = (E(i)-E_bar_run(i))/(DD_sine_squared(i));
121 end
122
123 alpha_sine_run = alpha_sine_run/T;
124
125 alpha_sine_run = transpose(alpha_sine_run);
126
127 alpha_sine_run(1) = 0;
128
129 alpha_sine_run(end) = 0;
130 %% Calculating E +alpha D run
131
132 E_D_sine_alpha_run = zeros([Steps,1]);
133 for i = 1:Steps
134 E_D_sine_alpha_run(i+1) = E_D_sine_alpha_run(i)+(E(i)+alpha_sine_run(i).*DD_sine
135 (i))*delta_t;
136 end
137
138 %Averaging E +ialphaD

```

```

136
137 for i = 1:Steps
138     E_D_sine_alpha_run_t(i) = sum(E_D_sine_alpha_run(i))/t(i);
139 end
140
141 E_D_sine_alpha_run_t = transpose(E_D_sine_alpha_run_t);
142
143 Sum_E_D_sine_alpha_run = sum(E_D_sine_alpha_run(Steps));
144 E_D_sine_alpha_run_bar = Sum_E_D_sine_alpha_run/T;
145
146
147 %% Standard Deviation of E+alpha D run
148
149 Std_alpha_run_sine = zeros([Steps,1]);
150 for i = 1:Steps
151     Std_alpha_run_sine(i) = (E_D_sine_alpha_run_t(i)-E_D_sine_alpha_run_bar);
152 end
153 Std_alpha_run_sine_sum = sum(Std_alpha_run_sine(Steps));
154 Std_alpha_run_sine_bar = Std_alpha_run_sine_sum/sqrt(T);
155
156
157 %% Variance of E+alpha D run
158 Var_alpha_run_sine = Std_alpha_run_sine.^2;
159 Var_alpha_run_sine_sum = Std_alpha_run_sine_sum.^2;
160 Var_alpha_run_sine_bar = Var_alpha_run_sine_sum/T;
161
162 %% Changing Minimum Time Average

```

### 7.3.6 Minimum Time Specific Auxiliary

```

1 %Function to calculate the Specific Auxillary Miniumum Time Case
2 %Created by Jamell Ivan Samuels
3
4 %% Finding the Specific Auxillary Function
5 %Calucldaing Lyapunov
6
7 %% Forming the Specific Lyapunov Auxillary Function
8
9 x1 = x(ind1:ind2,1);
10 x2 = x(ind1:ind2,2);
11 x3 = x(ind1:ind2,3);
12
13
14 MinT_DD_spec(:,1) = DD_spec(ind1:ind2,1);
15 MinT_DD_spec(:,2) = DD_spec(ind1:ind2,2);
16 MinT_DD_spec(:,3) = DD_spec(ind1:ind2,3);
17
18 %% Calculating Averages of DD
19 %% Average of DD
20 %Step Average of DD
21
22 %Integral of E + D
23 %Summation of D
24 MinT_DDSum_spec = zeros([Steps,1]);
25 for i = 1:Steps
26     MinT_DDSum_spec(i) = MinT_DD_spec(i,1)+MinT_DD_spec(i,2)+MinT_DD_spec(i,3);
27 end
28 MinT_DD_bar_spec = sum(MinT_DDSum_spec)/minT;
29 %Calculating D^2 Bar
30 MinT_DD_spec_squared= zeros([Steps,1]);
31 for i = 1:Steps

```

```

32     MinT_DD_spec_squared(i) = MinT_DDSum_spec(i).^2;
33 end
34 MinT_DD_spec_squared_Sum = sum(MinT_DD_spec_squared);
35 MinT_DD_spec_squared_bar = MinT_DD_spec_squared_Sum/minT; %DD Squared bar
36 %% Average of E & DD
37 %Combining of E & DD
38 MinT_Comb_E_D_spec = zeros([Steps,1]);
39 for i = 1:Steps
40     MinT_Comb_E_D_spec(i) = MinT_E(i)+MinT_DDSum_spec(i);
41 end
42
43 %Summation of E & DD
44 MinT_Sum_E_D_spec = zeros([Steps,1]);
45 for i = 1:Steps
46     MinT_Sum_E_D_spec(i+1) = (MinT_Sum_E_D_spec(i))+(MinT_Comb_E_D_spec(i)*
        delta_t);
47 end
48
49 %Average of E + D
50 for i = 1:Steps
51     MinT_E_D_spec_t(i) = sum(MinT_Sum_E_D_spec(i))/t_m(i);
52 end
53
54 MinT_E_D_spec_t = transpose(MinT_E_D_spec_t);
55 MinT_E_D_spec_bar = sum(MinT_Sum_E_D_spec(Steps))/minT; %Average of E + D
    specific
56
57
58 %% Variance of Specific Auxillary Minimum Time
59
60 for i = 1:Steps
61     MinT_Var_spec(i) = (MinT_E_D_spec_t(i) - MinT_E_D_spec_bar).^2;
62 end
63
64 MinT_Var_spec_bar = sum(MinT_Var_spec(Steps))/minT;
65
66 %% Calculation of alpha - General
67
68 for i = 1:Steps
69     disp(i)
70     MinT_alpha_spec(i) = (MinT_E(i)-MinT_E_bar)/(MinT_DD_spec_squared(i));
71 end
72
73 MinT_alpha_spec = MinT_alpha_spec/minT;
74
75 MinT_alpha_spec = transpose(MinT_alpha_spec);
76
77 %% Calculating Average of (E + alphaD) - Specific
78
79 %Combining E + alpha_bar D
80
81 MinT_E_D_spec_alpha = zeros([Steps,1]);
82 for i = 1:Steps
83     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;
84 end
85
86 %Averaging E +ialphaD
87
88 for i = 1:Steps

```

```

89     MinT_E_D_spec_alpha_t(i) = sum(MinT_E_D_spec_alpha(i))/t_m(i);
90 end
91 MinT_E_D_spec_alpha_t = transpose(MinT_E_D_spec_alpha_t);
92
93 MinT_Sum_E_D_spec_alpha = sum(MinT_E_D_spec_alpha(Steps));
94 MinT_E_D_spec_alpha_bar = MinT_Sum_E_D_spec_alpha/minT;
95
96 %% Standard Deviation of E+alpha_min D
97
98
99 MinT_Std_alpha_spec = zeros([Steps,1]);
100 for i = 1:Steps
101     MinT_Std_alpha_spec(i) = (MinT_E_D_spec_alpha_t(i)-MinT_E_D_spec_alpha_bar);
102 end
103 MinT_Std_alpha_spec_sum = sum(MinT_Std_alpha_spec(Steps));
104 MinT_Std_alpha_spec_bar = MinT_Std_alpha_spec_sum/minT;
105
106 %% Variance of E +alpha min D
107 MinT_Var_alpha_spec = MinT_Std_alpha_spec.^2;
108 MinT_Var_alpha_spec_sum = MinT_Std_alpha_spec_sum.^2;
109 MinT_Var_alpha_spec_bar = MinT_Var_alpha_spec_sum/minT;
110
111 %% Calculating Running Alpha
112
113 for i = 1:Steps
114     disp(i)
115     MinT_alpha_spec_run(i) = (MinT_E(i)-MinT_E_bar_run(i))/(MinT_DD_spec_squared
116         (i));
117 end
118
119 MinT_alpha_spec_run = MinT_alpha_spec_run/minT;
120
121 MinT_alpha_spec_run = transpose(MinT_alpha_spec_run);
122
123 %% Calculating E +alpha D run
124
125 MinT_E_D_spec_alpha_run = zeros([Steps,1]);
126 for i = 1:Steps
127     MinT_E_D_spec_alpha_run(i+1) = MinT_E_D_spec_alpha_run(i)+(MinT_E(i)+
128         MinT_alpha_spec_run(i).*(MinT_DD_Sum_spec(i)))*delta_t;
129 end
130
131 %%Averaging E +ialphaD
132
133 for i = 1:Steps
134     MinT_E_D_spec_alpha_run_t(i) = sum(MinT_E_D_spec_alpha_run(i))/t_m(i);
135 end
136
137 MinT_E_D_spec_alpha_run_t = transpose(MinT_E_D_spec_alpha_run_t);
138
139 MinT_Sum_E_D_spec_alpha_run = sum(MinT_E_D_spec_alpha_run(Steps));
140 MinT_E_D_spec_alpha_run_bar = MinT_Sum_E_D_spec_alpha_run/minT;
141
142 %% Standard Deviation of E+alpha D run
143
144 MinT_Std_alpha_run_spec = zeros([Steps,1]);
145 for i = 1:Steps
146     MinT_Std_alpha_run_spec(i) = (MinT_E_D_spec_alpha_run_t(i)-
147         MinT_E_D_spec_alpha_run_bar);

```



```

146 end
147 MinT_Std_alpha_run_spec_sum = sum(MinT_Std_alpha_run_spec(Steps));
148 MinT_Std_alpha_run_spec_bar = MinT_Std_alpha_run_spec_sum/sqrt(minT);
149
150
151 %% Variance of E+alpha D run
152 MinT_Var_alpha_run_spec = MinT_Std_alpha_run_spec.^2;
153 MinT_Var_alpha_run_spec_sum = MinT_Std_alpha_run_spec_sum.^2;
154 MinT_Var_alpha_run_spec_bar = MinT_Var_alpha_run_spec_sum/minT;
155
156 %% Changing Minimum Time Average

```

### 7.3.7 Minimum Time General Auxiliary

```

1  %Function to calculate the General AUxillary Lyapunov Function
2
3  %Created by Jamell Ivan Samuels
4
5
6  %% Forming the General Auxillary Lyapunov Function
7
8
9  x1 = x(:,1);
10 x2 = x(:,2);
11 x3 = x(:,3);
12
13
14 MinT_DD_gen(:,1) = dxdt(ind1:ind2);
15 MinT_DD_gen(:,2) = dydt(ind1:ind2);
16 MinT_DD_gen(:,3) = dzdt(ind1:ind2);
17 MinT_DD_gen(:,4) = x(ind1:ind2,1).*dydt(ind1:ind2) + x(ind1:ind2,2).*dxdt(ind1:
    ind2);
18 MinT_DD_gen(:,5) = x(ind1:ind2,1).*dzdt(ind1:ind2,1) + x(ind1:ind2,3).*dxdt(ind1
    :ind2);
19 MinT_DD_gen(:,6) = x(ind1:ind2,3).*dydt(ind1:ind2) + x(ind1:ind2,2).*dzdt(ind1:
    ind2);
20 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);
21 MinT_DD_gen(:,8) = 2.*x(ind1:ind2,1).*dxdt(ind1:ind2);
22 MinT_DD_gen(:,9) = 2.*x(ind1:ind2,2).*dydt(ind1:ind2);
23 MinT_DD_gen(:,10) = 2.*x(ind1:ind2,3).*dzdt(ind1:ind2);
24
25
26
27 %% Calculating Averages of DD
28 %% Average of DD
29 %Step Average of DD
30
31 %Integral of E + D
32 %Summation of D
33 MinT_DDSum_gen = zeros([Steps,1]);
34 for i = 1:Steps
35 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);
36 end
37 MinT_DD_bar_gen = sum(MinT_DDSum_gen)/minT;
38 %Calculating D^2 Bar
39 MinT_DD_gen_squared= zeros([Steps,1]);
40 for i = 1:Steps

```

```

41     MinT_DD_gen_squared(i) = (MinT_DDSum_gen(i)).^2;
42 end
43 MinT_DD_gen_squared_Sum = sum(MinT_DD_gen_squared);
44 MinT_DD_gen_squared_bar = MinT_DD_gen_squared_Sum/minT; %DD Squared bar
45 %% Average of E & DD
46 %Combining of E & DD
47 MinT_Comb_E_D_gen = zeros([Steps,1]);
48 for i = 1:Steps
49     MinT_Comb_E_D_gen(i) = E(i)+MinT_DDSum_gen(i);
50 end
51
52 %Summation of E & DD
53 MinT_Sum_E_D_gen = zeros([Steps,1]);
54 for i = 1:Steps
55     MinT_Sum_E_D_gen(i+1) = (MinT_Sum_E_D_gen(i))+(MinT_Comb_E_D_gen(i)*delta_t)
56     ;
57 end
58 %Average of E + D
59 for i = 1:Steps
60     MinT_E_D_gen_t(i) = sum(MinT_Sum_E_D_gen(i))/t_m(i);
61 end
62 MinT_E_D_gen_t = transpose(MinT_E_D_gen_t);
63
64
65 MinT_E_D_gen_bar = sum(MinT_Sum_E_D_gen(Steps))/minT; %Average of E + D specific
66
67
68 %% Variance of General Auxillary Minimum Time
69
70 for i = 1:Steps
71     MinT_Var_gen(i) = (MinT_E_D_gen_t(i) - MinT_E_D_gen_bar).^2;
72 end
73
74 MinT_Var_gen_bar = sum(MinT_Var_gen(Steps))/minT;
75
76
77
78
79 %% Calculation of alpha - Genaral
80
81 for i = 1:Steps
82     disp(i)
83     MinT_alpha_gen(i) = (MinT_E(i)-MinT_E_bar)/(MinT_DD_gen_squared(i));
84 end
85
86 MinT_alpha_gen = MinT_alpha_gen/minT;
87
88 MinT_alpha_gen = transpose(MinT_alpha_gen);
89
90 %% Calculating Average of (E + alpha_minD) - Specific
91
92 %Combining E + alpha D
93
94 MinT_E_D_gen_alpha = zeros([Steps,1]); %combs are e+alpha d
95 for i = 1:Steps
96     MinT_E_D_gen_alpha(i+1) = MinT_E_D_gen_alpha(i)+(MinT_E(i)+MinT_alpha_gen(i).*
97         MinT_DDSum_gen(i))*delta_t;
98 end
99

```

```

99 %Averaging E +alphaD
100
101 for i = 1:Steps
102     MinT_E_D_gen_alpha_t(i) = sum(MinT_E_D_gen_alpha(i))/t_m(i);
103 end
104 MinT_E_D_gen_alpha_t = transpose(MinT_E_D_gen_alpha_t);
105
106 MinT_Sum_E_D_gen_alpha = sum(MinT_E_D_gen_alpha(Steps));
107 MinT_E_D_gen_alpha_bar = MinT_Sum_E_D_gen_alpha/minT;
108
109 %% Calculating Runing Alpha - Specific
110
111 for i = 1:Steps
112     disp(i)
113     MinT_alpha_gen_run(i) = (E(i)-E_bar_run(i))/(DD_gen_squared(i));
114 end
115
116 MinT_alpha_gen_run = MinT_alpha_gen_run/minT;
117
118 %MinT_alpha_gen = transpose(MinT_alpha_gen)
119
120
121 %% Standard Deviation of E+alpha_min D
122
123 MinT_Std_alpha_gen = zeros([Steps,1]);
124 for i = 1:Steps
125     MinT_Std_alpha_gen(i) = (MinT_E_D_gen_alpha_t(i)-MinT_E_D_gen_alpha_bar);
126 end
127 MinT_Std_alpha_gen_sum = sum(MinT_Std_alpha_gen(Steps));
128 MinT_Std_alpha_gen_bar = MinT_Std_alpha_gen_sum/sqrt(minT);
129
130
131 %% Variance of E +alpha min D
132 MinT_Var_alpha_gen = MinT_Std_alpha_gen.^2;
133 MinT_Var_alpha_gen_sum = MinT_Std_alpha_gen_sum.^2;
134 MinT_Var_alpha_gen_bar = MinT_Var_alpha_gen_sum/minT;
135
136 %% Calculating Running Alpha
137
138 for i = 1:Steps
139     disp(i)
140     MinT_alpha_gen_run(i) = (MinT_E(i)-MinT_E_bar_run(i))/(MinT_DD_gen_squared(i)
141     ));
142 end
143
144 MinT_alpha_gen_run = MinT_alpha_gen_run/minT;
145
146 %MinT_alpha_gen_run = transpose(MinT_alpha_gen_run)
147
148 %% Calculating E +alpha D run
149
150 MinT_E_D_gen_alpha_run = zeros([Steps,1]);
151 for i = 1:Steps
152     MinT_E_D_gen_alpha_run(i+1) = MinT_E_D_gen_alpha_run(i)+(MinT_E(i)+
153     MinT_alpha_gen_run(i).*(MinT_DDSum_gen(i)))*delta_t;
154 end
155
156 %Averaging E +alphaD

```

```

157 for i = 1:Steps
158     MinT_E_D_gen_alpha_run_t(i) = sum(MinT_E_D_gen_alpha_run(i))/t_m(i);
159 end
160
161
162 MinT_E_D_gen_alpha_run_t = transpose(MinT_E_D_gen_alpha_run_t);
163
164 MinT_Sum_E_D_gen_alpha_run = sum(MinT_E_D_gen_alpha_run(Steps));
165 MinT_E_D_gen_alpha_run_bar = MinT_Sum_E_D_gen_alpha_run/minT;
166
167
168 %% Standard Deviation of E+alpha D run
169
170 MinT_Std_alpha_run_gen = zeros([Steps,1]);
171 for i = 1:Steps
172     MinT_Std_alpha_run_gen(i) = (MinT_E_D_gen_alpha_run_t(i)-
173         MinT_E_D_gen_alpha_run_bar);
174 end
175 MinT_Std_alpha_run_gen_sum = sum(MinT_Std_alpha_run_gen(Steps));
176 MinT_Std_alpha_run_gen_bar = MinT_Std_alpha_run_gen_sum/sqrt(minT);
177
178 %% Variance of E+alpha D run
179 MinT_Var_alpha_run_gen = MinT_Std_alpha_run_gen.^2;
180 MinT_Var_alpha_run_gen_sum = MinT_Std_alpha_run_gen_sum.^2;
181 MinT_Var_alpha_run_gen_bar = MinT_Var_alpha_run_gen_sum/minT;
182
183
184 %% Changing Minimum Time Average

```

### 7.3.8 Minimum Time Sine Auxiliary

```

1 %Function to calculate the Minimum Time Sine Auxillary Case
2 %Created by Jamell Ivan Samuels
3
4 %% Finding the Specific Auxillary Function
5
6 %% Forming the Sine Auxillary Function
7
8
9 MinT_DD_sine = DD_sine(ind1:ind2);
10
11
12 %% Calculating Averages of DD
13 %% Average of DD
14 %%Step Average of DD
15
16 %Integral of E + D
17 %Summation of D
18 MinT_DDSum_sine = sum(MinT_DD_sine);
19
20 MinT_DD_bar_sine = sum(MinT_DDSum_sine)/minT;
21 %Calculating D^2 Bar
22 for i = 1:Steps
23     MinT_DD_sine_squared(i) = (MinT_DD_sine(i)).^2;
24 end
25 MinT_DD_sine_squared_Sum = sum(MinT_DD_sine_squared);
26 MinT_DD_sine_squared_bar = MinT_DD_sine_squared_Sum/minT; %DD Squared bar
27 %% Average of E & DD
28 %Combining of E & DD
29 MinT_Comb_E_D_sine = zeros([Steps,1]);

```

```

30 for i = 1:Steps
31 MinT_Comb_E_D_sine(i) = MinT_E(i)+MinT_DD_sine(i);
32 end
33
34 %Summation of E & DD
35 MinT_Sum_E_D_sine = zeros([Steps,1]);
36 for i = 1:Steps
37     MinT_Sum_E_D_sine(i+1) = (MinT_Sum_E_D_sine(i))+(MinT_Comb_E_D_sine(i)*
        delta_t);
38 end
39
40 %Average of E + D
41
42 for i = 1:Steps
43     MinT_E_D_sine_t(i) = sum(MinT_Sum_E_D_sine(i))/t_m(i);
44 end
45 MinT_E_D_sine_t = transpose(MinT_E_D_sine_t);
46
47
48 MinT_E_D_sine_bar = sum(MinT_Sum_E_D_sine(Steps))/minT; %Average of E + D
    specific
49
50
51 %% Variance of Sine Auxillary Minimum Time
52
53 for i = 1:Steps
54     MinT_Var_sine(i) = (MinT_E_D_sine_t(i) - MinT_E_D_sine_bar).^2;
55 end
56
57 MinT_Var_sine_bar = sum(MinT_Var_sine(Steps))/minT;
58
59
60
61 %% Calculation of alpha min - Sine
62
63 for i = 1:Steps
64     disp(i)
65     MinT_alpha_sine(i) = (MinT_E(i)-MinT_E_bar)/(MinT_DD_sine_squared(i));
66 end
67
68 MinT_alpha_sine = MinT_alpha_sine/minT;
69
70 MinT_alpha_sine = transpose(MinT_alpha_sine);
71
72
73 %% Calculating Average of (E + alphaD) - Sine
74
75 %Combining E + alpha_bar D
76
77 MinT_E_D_sine_alpha = zeros([Steps,1]);
78 for i = 1:Steps
79     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;
80 end
81
82 %Averaging E +ialphaD
83
84 for i = 1:Steps
85     MinT_E_D_sine_alpha_t(i) = sum(MinT_E_D_sine_alpha(i))/t_m(i);
86 end

```

```

87 MinT_E_D_sine_alpha_t = transpose(MinT_E_D_sine_alpha_t);
88
89 MinT_Sum_E_D_sine_alpha = sum(MinT_E_D_sine_alpha(Steps));
90 MinT_E_D_sine_alpha_bar = MinT_Sum_E_D_sine_alpha/minT;
91
92 %% Standard Deviation of E +alpha D
93
94 MinT_Std_alpha_sine = zeros([Steps,1]);
95 for i =1:Steps
96     MinT_Std_alpha_sine(i) = (MinT_E_D_sine_alpha_t(i)-MinT_E_D_sine_alpha_bar);
97     % This is is the sample deviation
98 end
99 MinT_Std_alpha_sine_sum = sum(MinT_Std_alpha_sine(Steps));
100 MinT_Std_alpha_min_sine_bar = MinT_Std_alpha_sine_sum/sqrt(minT); %This is the
    standard deviation
101
102 %% Variance of E+alpha D
103 MinT_Var_alpha_sine = MinT_Std_alpha_sine.^2;
104 MinT_Var_alpha_sine_sum = MinT_Std_alpha_sine_sum.^2;
105 MinT_Var_alpha_sine_bar = MinT_Var_alpha_sine_sum/minT;
106
107 %% Calculating Running Alpha
108
109 for i = 1:Steps
110     disp(i)
111     MinT_alpha_sine_run(i) = (MinT_E(i)-MinT_E_bar_run(i))/(MinT_DD_sine_squared
        (i));
112 end
113
114 MinT_alpha_sine_run = MinT_alpha_sine_run/minT;
115
116 MinT_alpha_sine_run = transpose(MinT_alpha_sine_run);
117
118 %% Calculating E +alpha D run
119
120 MinT_E_D_sine_alpha_run = zeros([Steps,1]);
121 for i = 1:Steps
122     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;
123 end
124
125 %Averaging E +ialphaD
126
127 for i = 1:Steps
128     MinT_E_D_sine_alpha_run_t(i) = sum(MinT_E_D_sine_alpha_run(i))/t_m(i);
129 end
130
131 MinT_E_D_sine_alpha_run_t = transpose(MinT_E_D_sine_alpha_run_t);
132
133 MinT_Sum_E_D_sine_alpha_run = sum(MinT_E_D_sine_alpha_run(Steps));
134 MinT_E_D_sine_alpha_run_bar = MinT_Sum_E_D_sine_alpha_run/minT;
135
136
137 %% Standard Deviation of E+alpha D run
138
139 MinT_Std_alpha_run_sine = zeros([Steps,1]);
140 for i =1:Steps
141     MinT_Std_alpha_run_sine(i) = (MinT_E_D_sine_alpha_run_t(i)-
        MinT_E_D_sine_alpha_run_bar);

```

```

142 end
143 MinT_Std_alpha_run_sine_sum = sum(MinT_Std_alpha_run_sine(Steps));
144 MinT_Std_alpha_run_sine_bar = MinT_Std_alpha_run_sine_sum/sqrt(minT);
145
146
147 %% Variance of E+alpha D run
148 MinT_Var_alpha_run_sine = MinT_Std_alpha_run_sine.^2;
149 MinT_Var_alpha_run_sine_sum = MinT_Std_alpha_run_sine_sum.^2;
150 MinT_Var_alpha_run_sine_bar = MinT_Var_alpha_run_sine_sum/minT;
151
152 %% Changing Minimum Time Average

```

### 7.3.9 Plot Function

```

1 % Function created to plot diagrams
2 %Created by Jamell Ivan Samuels
3
4 %% Plotting All Averages
5 %Specific
6 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], '-. ');
7 xlabel('time (s)')
8 u = ylabel('Quantity of interest ($$\overline{\Phi}$$)')
9 set(u, 'Interpreter', 'latex', 'fontsize', 12)
10 h = legend('$$\overline{E_{\{true\}}}$$', '$$\overline{E}$$', '$$\overline{E+D}$$', '$$
    \overline{E+\alpha_{\min}D}$$', '$$\overline{E+\alpha_{\text{run}}D}$$', 'Location', '
    Best')
11 set(h, 'Interpreter', 'latex', 'fontsize', 12)
12 print(gcf, 'Specific_Averages.png', '-dpng')
13 %General
14 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], '-. ');
15 xlabel('time (s)')
16 u = ylabel('Quantity of interest ($$\overline{\Phi}$$)')
17 set(u, 'Interpreter', 'latex', 'fontsize', 12)
18 h = legend('$$\overline{E_{\{true\}}}$$', '$$\overline{E+D}$$', '$$\overline{E+\alpha_{
    \min}D}$$', '$$\overline{E+\alpha_{\text{run}}D}$$', 'Location', 'Best')
19 set(h, 'Interpreter', 'latex', 'fontsize', 12)
20 print(gcf, 'General_Averages.png', '-dpng')
21 %Sine
22 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], '-. ');
23 xlabel('time (s)')
24 u = ylabel('Quantity of interest ($$\overline{\Phi}$$)')
25 set(u, 'Interpreter', 'latex', 'fontsize', 12)
26 h = legend('$$\overline{E_{\{true\}}}$$', '$$\overline{E}$$', '$$\overline{E+D}$$', '$$
    \overline{E+\alpha_{\min}D}$$', '$$\overline{E+\alpha_{\text{run}}D}$$', 'Location', '
    Best')
27 set(h, 'Interpreter', 'latex', 'fontsize', 12)
28 print(gcf, 'Sine_Averages.png', '-dpng')
29
30 %% Plot Side by Side Average Comparison
31 %Base Systems
32 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], '-. ');
33 xlabel('time (s)')
34 u = ylabel('Quantity of interest ($$\overline{\Phi}$$)')
35 set(u, 'Interpreter', 'latex', 'fontsize', 12)
36 legend('True', 'Base', 'Specific', 'General', 'Sine', 'Location', 'Best')
37 print(gcf, 'Base_Averages.png', '-dpng')
38

```

```

39 %alpha Systems
40 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], '-. ');
41 xlabel('time (s)')
42 u = ylabel('Quantity of interest ( $\overline{\Phi}$ )')
43 set(u, 'Interpreter', 'latex', 'fontsize', 12)
44 legend('True', 'Specific', 'General', 'Sine', 'Location', 'Best')
45 print(gcf, 'Alpha_Averages.png', '-dpng')
46
47 %alpha run Systems
48
49 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], '-. ');
50 xlabel('time (s)')
51 u = ylabel('Quantity of interest ( $\overline{\Phi}$ )')
52 legend('True', 'Specific', 'General', 'Sine', 'Location', 'Best')
53 set(u, 'Interpreter', 'latex', 'fontsize', 12)
54 print(gcf, 'Alpha_Run_Averages.png', '-dpng')
55
56 %% Plot All Variances
57
58 %Specific
59 plot(t,[0;Var], ':',t,[0;Var_spec], '—',t,[0;Var_alpha_spec], '-.',t,[0;
      Var_alpha_run_spec]);
60 xlabel('time (s)')
61 ylabel('Variance ( $\sigma^2$ )')
62 legend('\sigma^2', '\sigma_{spec}^2', '\sigma_{spec alpha}^2', '\sigma_{spec alpha
      run}^2', 'Location', 'Best')
63 print(gcf, 'Specific_Variances.png', '-dpng')
64 %General
65 plot(t,[0;Var], ':',t,[0;Var_gen], '—',t,[0;Var_alpha_gen], '-.',t,[0;
      Var_alpha_run_gen]);
66 xlabel('time (s)')
67 ylabel('Variance ( $\sigma^2$ )')
68 legend('\sigma^2', '\sigma_{gen}^2', '\sigma_{gen alpha}^2', '\sigma_{gen alpha run
      }^2', 'Location', 'Best')
69 print(gcf, 'General_Variances.png', '-dpng')
70 %Sine
71 plot(t,[0;Var], ':',t,[0;Var_sine], '—',t,[0;Var_alpha_sine], '-.',t,[0;
      Var_alpha_run_sine]);
72 xlabel('time (s)')
73 ylabel('Variance ( $\sigma^2$ )')
74 legend('\sigma^2', '\sigma_{sine}^2', '\sigma_{sine alpha}^2', '\sigma_{sine alpha
      run}^2', 'Location', 'Best')
75 print(gcf, 'Sine_Variances.png', '-dpng')
76
77
78
79 %% Plot Side by Side Variance Comparison
80 %Base
81 plot(t,[0;Var], '—',t,[0;Var_spec], ':',t,[0;Var_gen], '-.',t,[0;Var_sine]);
82 xlabel('time (s)')
83 ylabel('Variance ( $\sigma^2$ )')
84 legend('\sigma^2', '\sigma_{spec}^2', '\sigma_{gen}^2', '\sigma_{sine}^2', 'Location
      ', 'Best')
85 print(gcf, 'Base_Variances.png', '-dpng')
86
87 %alpha
88 plot(t,[0;Var], '—',t,[0;Var_alpha_spec], ':',t,[0;Var_alpha_gen], '-.',t,[0;
      Var_alpha_sine]);

```



```

89 xlabel('time (s)')
90 ylabel('Variance (\sigma^2)')
91 legend('\sigma^2', '\sigma_{spec}^2', '\sigma_{gen}^2', '\sigma_{sine}^2', 'Location', 'Best')
92 print(gcf, 'Alpha-Variances.png', '-dpng')
93
94 %alpha run
95 plot(t,[0;Var], '—', t,[0;Var_alpha_run_spec], ':', t,[0;Var_alpha_run_gen], '-.', t,[0;Var_alpha_run_sine]);
96 xlabel('time (s)')
97 ylabel('Variance \sigma^2')
98 legend('\sigma^2', '\sigma_{spec}^2', '\sigma_{gen}^2', '\sigma_{sine}^2', 'Location', 'Best')
99 print(gcf, 'Alpha-Run-Variances_.png', '-dpng')
100
101 %% Plot Side by Side alpha vs alpha run comparison
102 %all alphas
103
104 plot(t,[0;alpha_spec], '—', t,[0;alpha_gen], ':', t,[0;alpha_sine], '-. ');
105 xlabel('time (s)')
106 ylabel('\alpha')
107 legend('\alpha_{spec}', '\alpha_{gen}', '\alpha_{sine}', 'Location', 'Best')
108 print(gcf, 'Alpha.png', '-dpng')
109
110
111 %all alpha runs
112 plot(t,[0;alpha_spec_run], '—', t,[0;alpha_gen_run], ':', t,[0;alpha_sine_run], '-. ');
113 xlabel('time (s)')
114 ylabel('\alpha')
115 legend('\alpha_{spec}', '\alpha_{gen}', '\alpha_{sine}', 'Location', 'Best')
116 print(gcf, 'Alpha-Run.png', '-dpng')
117
118 %specs
119
120 plot(t,[0;alpha_spec], '—', t,[0;alpha_spec_run], ':');
121 xlabel('time (s)')
122 ylabel('\alpha')
123 legend('\alpha_{spec}', '\alpha_{spec_run}', 'Location', 'Best')
124 print(gcf, 'Alpha-Spec_.png', '-dpng')
125
126 %gens
127 plot(t,[0;alpha_gen], '—', t,[0;alpha_gen_run], ':');
128 xlabel('time (s)')
129 ylabel('\alpha')
130 legend('\alpha_{gen}', '\alpha_{gen_run}', 'Location', 'Best')
131 print(gcf, 'Alpha-Gen_.png', '-dpng')
132
133 %sines
134
135 plot(t,[0;alpha_sine], '—', t,[0;alpha_sine_run], ':');
136 xlabel('time (s)')
137 ylabel('\alpha')
138 legend('\alpha_{sine}', '\alpha_{sine_run}', 'Location', 'Best')
139 print(gcf, 'Alpha-Sine_.png', '-dpng')

```

### 7.3.10 Minimum Time Plot Function

```

1 % Function created to plot diagrams
2 %Created by Jamell Ivan Samuels
3

```

```

4 %% Plotting All Averages
5 %Specific
6 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], '
    -');
7 xlabel('time (s)')
8 u = ylabel('Quantity of interest ($$\overline{\Phi}$$)')
9 set(u, 'Interpreter', 'latex', 'fontsize', 12)
10 h = legend('$$\overline{E_{\{true\}}}$$', '$$\overline{E}$$', '$$\overline{E+D}$$', '$$
    \overline{E+\alpha_{\min}D}$$', '$$\overline{E+\alpha_{\text{run}}D}$$', 'Location', '
    Best')
11 set(h, 'Interpreter', 'latex', 'fontsize', 12)
12 print(gcf, 'MinT_Specific_Averages.png', '-dpng')
13 %General
14 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], '-');
15 ;
16 xlabel('time (s)')
17 u = ylabel('Quantity of interest ($$\overline{\Phi}$$)')
18 set(u, 'Interpreter', 'latex', 'fontsize', 12)
19 h = legend('$$\overline{E_{\{true\}}}$$', '$$\overline{E+D}$$', '$$\overline{E+\alpha_{\min}D}$$', '$$
    \overline{E+\alpha_{\text{run}}D}$$', 'Location', 'Best')
20 set(h, 'Interpreter', 'latex', 'fontsize', 12)
21 print(gcf, 'MinT_General_Averages.png', '-dpng')
22 %Sine
23 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], '
    -');
24 xlabel('time (s)')
25 u = ylabel('Quantity of interest ($$\overline{\Phi}$$)')
26 set(u, 'Interpreter', 'latex', 'fontsize', 12)
27 h = legend('$$\overline{E_{\{true\}}}$$', '$$\overline{E}$$', '$$\overline{E+D}$$', '$$
    \overline{E+\alpha_{\min}D}$$', '$$\overline{E+\alpha_{\text{run}}D}$$', 'Location', '
    Best')
28 set(h, 'Interpreter', 'latex', 'fontsize', 12)
29 print(gcf, 'MinT_Sine_Averages.png', '-dpng')
30 %% Plot Side by Side Average Comparison
31 %Base Systems
32 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], '-');
33 xlabel('time (s)')
34 u = ylabel('Quantity of interest ($$\overline{\Phi}$$)')
35 set(u, 'Interpreter', 'latex', 'fontsize', 12)
36 legend('True', 'Base', 'Specific', 'General', 'Sine', 'Location', 'Best')
37 print(gcf, 'MinT_Base_Averages.png', '-dpng')
38
39 %alpha Systems
40 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], '-');
41 xlabel('time (s)')
42 u = ylabel('Quantity of interest ($$\overline{\Phi}$$)')
43 set(u, 'Interpreter', 'latex', 'fontsize', 12)
44 legend('True', 'Specific', 'General', 'Sine', 'Location', 'Best')
45 print(gcf, 'MinT_Alpha_Averages.png', '-dpng')
46
47 %alpha run Systems
48
49 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], '-');

```

```

50 xlabel('time (s)')
51 u = ylabel('Quantity of interest ($$\overline{\Phi}$$)')
52 legend('True','Specific','General','Sine','Location','Best')
53 set(u,'Interpreter','latex','fontsize',12)
54 print(gcf,'MinT_Alpha_Run_Averages.png','-dpng')
55
56 %% Plot All Variances
57
58 %Specific
59 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]);
60 xlabel('time (s)')
61 ylabel('Variance (\sigma^2)')
62 legend('\sigma^2','\sigma_{spec}^2','\sigma_{spec alpha}^2','\sigma_{spec alpha run}^2','Location','Best')
63 print(gcf,'MinT_Specific_Variances.png','-dpng')
64 %General
65 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]);
66 xlabel('time (s)')
67 ylabel('Variance (\sigma^2)')
68 legend('\sigma^2','\sigma_{gen}^2','\sigma_{gen alpha}^2','\sigma_{gen alpha run}^2','Location','Best')
69 print(gcf,'MinT_General_Variances.png','-dpng')
70 %Sine
71 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]);
72 xlabel('time (s)')
73 ylabel('Variance (\sigma^2)')
74 legend('\sigma^2','\sigma_{sine}^2','\sigma_{sine alpha}^2','\sigma_{sine alpha run}^2','Location','Best')
75 print(gcf,'MinT_Sine_Variances.png','-dpng')
76
77
78
79 %% Plot Side by Side Variance Comparison
80 %Base
81 plot(t_m,[0;MinT_Var],':',t_m,[0;MinT_Var_spec],'—',t_m,[0;MinT_Var_gen],'-.',
    t_m,[0;MinT_Var_sine]);
82 xlabel('time (s)')
83 ylabel('Variance (\sigma^2)')
84 legend('\sigma^2','\sigma_{spec}^2','\sigma_{gen}^2','\sigma_{sine}^2','Location',
    ',','Best')
85 print(gcf,'MinT_Base_Variances.png','-dpng')
86
87 %alpha
88 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]);
89 xlabel('time (s)')
90 ylabel('Variance (\sigma^2)')
91 legend('\sigma^2','\sigma_{spec}^2','\sigma_{gen}^2','\sigma_{sine}^2','Location',
    ',','Best')
92 print(gcf,'MinT_Alpha_Variances.png','-dpng')
93
94 %alpha run
95 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]);
96 xlabel('time (s)')
97 ylabel('Variance (\sigma^2)')
98 legend('\sigma^2','\sigma_{spec}^2','\sigma_{gen}^2','\sigma_{sine}^2','Location

```

```

    ', 'Best')
99 print(gcf, 'MinT_Alpha_Run_VariANCES_.png', '-dpng')
100
101 %% Plot Side by Side alpha vs alpha run comparison
102 %all alphas
103
104 plot(t_m,[0; MinT_alpha_spec], ':', t_m,[0; MinT_alpha_gen], '—', t_m,[0;
    MinT_alpha_sine], '-. ');
105 xlabel('time (s)')
106 ylabel('\alpha')
107 legend('\alpha', '\alpha_{spec}', '\alpha_{gen}', '\alpha_{sine}', 'Location', 'Best'
    )
108 print(gcf, 'MinT_Alpha.png', '-dpng')
109
110
111 %all alpha runs
112 plot(t_m,[0; MinT_alpha_spec_run], ':', t_m,[0; MinT_alpha_gen_run], '—', t_m,[0;
    MinT_alpha_sine_run], '-. ');
113 xlabel('time (s)')
114 ylabel('\alpha')
115 legend('\alpha_{spec}', '\alpha_{gen}', '\alpha_{sine}', 'Location', 'Best')
116 print(gcf, 'MinT_Alpha_Run.png', '-dpng')
117
118 %specs
119
120 plot(t_m,[0; MinT_alpha_spec], ':', t_m,[0; MinT_alpha_spec_run], '—');
121 xlabel('time (s)')
122 ylabel('\alpha')
123 legend('\alpha_{spec}', '\alpha_{spec_run}', 'Location', 'Best')
124 print(gcf, 'MinT_Alpha_Spec_.png', '-dpng')
125
126 %gens
127 plot(t_m,[0; MinT_alpha_gen], ':', t_m,[0; MinT_alpha_gen_run], '—');
128 xlabel('time (s)')
129 ylabel('\alpha')
130 legend('\alpha_{gen}', '\alpha_{gen_run}', 'Location', 'Best')
131 print(gcf, 'MinT_Alpha_Gen_.png', '-dpng')
132
133 %sines
134
135 plot(t_m,[0; MinT_alpha_sine], ':', t_m,[0; MinT_alpha_sine_run], '—');
136 xlabel('time (s)')
137 ylabel('\alpha')
138 legend('\alpha_{sine}', '\alpha_{sine_run}', 'Location', 'Best')
139 print(gcf, 'MinT_Alpha_Sine_.png', '-dpng')

```