Parallel Non-Monte Carlo Transient Noise Simulation

Alex Goulet

Master of Engineering

Department of Electrical and Computer Engineering

McGill University

Montréal, Québec

2021-12-13

A thesis submitted to McGill University in partial of the requirements of the degree of Master of Engineering in Electrical Engineering

DEDICATION

To my father. May he always watch over me and be proud.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my research supervisor, Prof. Roni Khazaka. I am immensely grateful for his guidance, encouragement, and confidence in my abilities from day one. His positive influence on my research as well as myself constitutes a major factor in the sense of fulfillment that I have experienced throughout my Master's program.

I would also like to express my gratitude to my friends Karanvir Sidhu and Marco Kassis from the electronic design automation group for helping me tremendously with my research and making me feel at home in the lab.

My deepest thanks go to Dr. Mina Farhan from Cadence Design System, Inc. for his mentorship and vital contribution to this thesis.

Last but not least, I would like to thank my girlfriend Nicole Andersen and her parents, Dr. Ross Andersen and Dr. Susan Bartlett, for their unconditional support over the past few years.

ABSTRACT

Noise analysis is a necessary but often lengthy part of the design process for microelectronics. Frequency domain noise analysis methods are ill-suited for simulating noise in nonlinear circuits with arbitrary large-signal waveforms, while time domain noise analysis methods typically suffer from long simulation times. A novel parallel non-Monte Carlo transient noise analysis method for general nonlinear analysis is presented. Non-Monte Carlo based transient noise methods avoid expensive and lengthy Monte Carlo iterations by describing the nonlinear system with a linear time-varying system that can be used to calculate the variance of the waveform at every time point. However, the main bottleneck in these methods is that the linear time-varying system is computationally expensive to solve when the circuit is large. To address this issue, the proposed method decouples the linear time-varying system into homogeneous and inhomogeneous systems that can be solved independently. The inhomogeneous system is solved in parallel by partitioning the time interval into independent subintervals, while the complexity of the homogeneous system can be decreased significantly due to the linearity of the equations.

RÉSUMÉ

L'analyse de bruit est une étape nécessaire mais longue du processus de conception de la microélectronique. Les méthodes d'analyse de bruit dans le domain fréquentiel sont mal adaptées à la simulation de bruit de circuits non linéaires avec formes d'ondes à grand signal, tandis que les méthodes d'analyse de bruit dans le domain temporel souffrent typiquement de longs temps de simulation. Une nouvelle méthode de simulation transitoire non Monte-Carlo de bruit en parallèle pour analysis générale non linéaire est présentée. Les méthodes de simulation transitoire de bruit qui ne sont pas basées sur la méthode de Monte-Carlo évitent les itérations longues et dispendieuses de la méthode de Monte-Carlo en décrivant le système non linéaire à l'aide d'un système linéaire variable dans le temps qui peut être utilisé pour calculer la variance de la forme d'onde à chaque point dans le temps. Cependant, le principal goulot avec ces méthodes est que le système linéaire variable dans le temps est coûteux à résoudre en ressources informatiques lorsque le circuit est large. Afin de résoudre ce problème, la méthode proposée découple le système linéaire variable dans le temps en systèmes homogènes et non homogènes qui peuvent être résolus indépendamment. Le système non homogène est résolu en parallèle en partitionnant l'intervalle de temps en sous-intervalles indépendants, tandis que la complexité du système homogène peut être réduite de façon significative en raison de la linéarité des équations.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AC	**A**lternating **C**urrent
BJT	**B**ipolar **J**unction **T**ransistor
CPU	**C**entral **P**rocessing **U**nit
DC	**D**irect **C**urrent
GMRES	**G**eneralized **M**inimal **Res**idual Method
KCL	**K**irchoff's **C**urrent **L**aw
LNA	**L**ow-**N**oise **A**mplifier
LPTV	**L**inear **P**eriodically **T**ime-**V**arying
LTI	**L**inear **T**ime-**I**nvariant
LTV	**L**inear **T**ime-**V**arying
LU	**L**ower-**U**pper Triangular Decomposition
MNA	**M**odified **N**odal **A**nalysis
PSD	**P**ower **S**pectral **D**ensity
RF	**R**adio **F**requency
RMS	**R**oot **M**ean **S**quare

CHAPTER 1
Introduction

1.1 Background and Motivation

Noise analysis poses a significant problem in the design of microwave and radio frequency (RF) circuits. Electrical noise in circuits is inherent to the circuit components and manifests itself as random fluctuations in the amplitude and phase of any signal [19]. Although small, these fluctuations can nonetheless impact the behavior of integrated circuits and must be taken into account in the early design stages. For example, the noise produced internally by a low-noise amplifier or mixer circuit in a RF front end which receives very weak signals as input must be accounted for in order to preserve the quality of the signal [17].

Noise analysis methods can be categorized into frequency and time domain based methods. In frequency domain methods, the circuit is linearized around its equilibrium point (either DC or periodic state) [22]. The total output noise is obtained by first computing the transfer functions between each noise source and the output. The total noise is then calculated by multiplying the source noise power with its transfer function. However, such an approach is limited to small-signal analysis around an equilibrium point. The noise in a mixer circuit, for instance, would not be simulated adequately using frequency domain methods due to the presence of a large local oscillator signal which causes the operating points of the active devices to change significantly over time [8].

For large-signal noise analysis of nonlinear circuits, time domain methods are desirable. Time domain methods are traditionally predicated on the Monte

Carlo method. In Monte Carlo based noise analysis methods, many transient analyses are performed to represent different paths taken by the noise sources. The noise sources are generated using a random number generator at every time point and the statistics of the noise are then computed across the Monte Carlo simulations. However, the main difficulty in this approach is its large central processing unit (CPU) cost.

Another group of time domain methods that is based on the theory of stochastic differential equations (SDEs) avoids many drawbacks of Monte Carlo based methods [7, 23]. These methods linearize the system of SDEs which governs the noise in the circuit at every time point. The noise correlation matrix can then be directly modeled using a linear time-varying (LTV) system. While this approach is typically faster than the Monte Carlo method, it can still be computationally expensive when the circuit is large due to the need to solve a Lyapunov equation at every time point.

In this thesis, a novel parallel non-Monte Carlo transient noise analysis method is presented [11]. The proposed method is inspired by the *paraexp* algorithm and reformulates the LTV system at the core of a non-Monte Carlo transient noise analysis method into homogeneous and inhomogeneous systems that can be solved independently [9]. By partitioning the time interval of interest into several independent subintervals, the inhomogeneous system can be solved in parallel for each subinterval. Furthermore, the homogeneous system can be solved efficiently by taking advantage of the linearity of the system. By solving the most computationally expensive step in parallel, the proposed method effectively reduces the overall CPU cost of the noise simulation.

1.2 Contribution

The original contribution presented in this thesis corresponds to a novel parallel non-Monte Carlo transient noise analysis method that aims to reduce transient noise simulation time by implementing a non-Monte Carlo transient noise analysis method in parallel. The details of this method are found in Chapter 4.

1.3 Organization of Thesis

This thesis is structured as follows. Chapter 2 first lays the foundation for noise analysis by covering the formulation of the system which models a given circuit without noise as well as some circuit simulation techniques to solve it, such as direct current (DC) analysis, linear multistep methods, and small-signal alternating current (AC) analysis. Chapter 3 then builds upon Chapter 2 by adding noise to the formulation of the system. Noise models as well as other concepts related to noise are discussed, followed by a description of noise analysis methods which augment the circuit simulation techniques from Chapter 2 to allow them to solve stochastic noisy systems rather than deterministic noiseless ones. Afterward, Chapter 4 presents a novel parallel noise analysis method that is based on a parallel implementation of one of the noise analysis methods covered in Chapter 3. A numerical example with results is also provided in Chapter 4 to compare the performance of the proposed method with that of other noise analysis methods. Finally, Chapter 5 provides a summary of the thesis along with some potential future work to be done on the topic of the presented research.

CHAPTER 2
Background on Circuit Simulation

2.1 Modified Nodal Analysis Formulation

The noise simulation problem will be formulated in this thesis using the Modified Nodal Analysis (MNA) formulation for the compact representation of circuit equations [13]. The MNA equations for any general circuit can be expressed as

$$\boldsymbol{F}(\dot{\boldsymbol{x}}(t), \boldsymbol{x}(t), t) = 0 \qquad \boldsymbol{x}(0) = \boldsymbol{x}_0, \tag{2.1}$$

where $\boldsymbol{x}(t) \in \mathbb{R}^n$ is the vector of circuit variables, $\dot{\boldsymbol{x}}(t)$ is the time derivative of $\boldsymbol{x}(t)$, and $\boldsymbol{F}$ maps $\boldsymbol{x}(t)$, $\dot{\boldsymbol{x}}(t)$, and t to a vector of real numbers of dimension n. In the MNA formulation, the circuit variables comprise the node voltages and branch currents obtained through the formulation of the node equations using Kirchoff's Current Law (KCL).

From (2.1), the MNA equations for an arbitrary circuit which contains both linear and nonlinear elements can be further developed into the following formulation:

$$\boldsymbol{G}\,\boldsymbol{x}(t) + \boldsymbol{C}\,\dot{\boldsymbol{x}}(t) + \boldsymbol{f}(\boldsymbol{x}(t)) = \boldsymbol{b}(t) \qquad \boldsymbol{x}(0) = \boldsymbol{x}_0, \tag{2.2}$$

where $\boldsymbol{G} \in \mathbb{R}^{n \times n}$ is a constant matrix which contains the contributions from the linear lumped memoryless elements such as resistors, $\boldsymbol{C} \in \mathbb{R}^{n \times n}$ is a constant matrix which contains the contributions from the linear lumped memory elements such as capacitors, $\boldsymbol{f}(\boldsymbol{x}(t)) \in \mathbb{R}^n$ is a vector of algebraic functions which accounts for the contributions of the nonlinear circuit elements, and $\boldsymbol{b}(t) \in \mathbb{R}^n$ is a vector which contains the contributions from the independent

current and voltage sources. All entries in $\boldsymbol{G}$, $\boldsymbol{C}$, and $\boldsymbol{b}(t)$ can be automatically generated from the circuit netlist using preexisting component stencils, which capture the contributions from the circuit components and express them in a single system of equations [30].

As an example of how (2.2) would be implemented for a given circuit, consider the simple nonlinear circuit shown in Fig. 2–1. Applying KCL at the three nodes in the circuit while also adding an additional equation for the relationship between the inductor voltage and current yields the following set of equations:

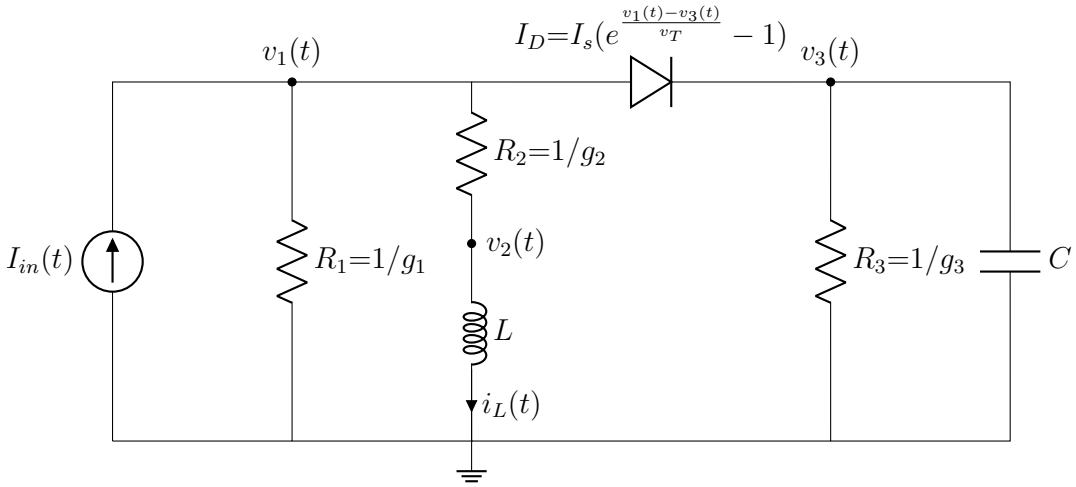

Figure 2–1: Sample nonlinear circuit.

$$g_1\, v_1(t) + g_2(v_1(t) - v_2(t)) + I_s\left(e^{\frac{v_1(t)-v_3(t)}{v_T}} - 1\right) = I_{in}(t), \tag{2.3a}$$

$$g_2(v_2(t) - v_1(t)) + i_L(t) = 0, \tag{2.3b}$$

$$g_3\, v_3(t) + C\,\dot{v}_3(t) - I_s\left(e^{\frac{v_1(t)-v_3(t)}{v_T}} - 1\right) = 0, \tag{2.3c}$$

$$v_2(t) - L\,\dot{i}_L = 0. \tag{2.3d}$$

Rewriting the set of equations in (2.3) in the format of the matrix equation in (2.2) results in the following system of equations:

$$\underbrace{\begin{bmatrix} g_1+g_2 & -g_2 & 0 & 0 \\ -g_2 & g_2 & 0 & 1 \\ 0 & 0 & g_3 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}}_{\boldsymbol{G}} \underbrace{\begin{bmatrix} v_1(t) \\ v_2(t) \\ v_3(t) \\ i_L(t) \end{bmatrix}}_{\boldsymbol{x}(\mathrm{t})} + \underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & C & 0 \\ 0 & 0 & 0 & -L \end{bmatrix}}_{\boldsymbol{C}} \underbrace{\begin{bmatrix} \dot{v}_1(t) \\ \dot{v}_2(t) \\ \dot{v}_3(t) \\ \dot{i}_L(t) \end{bmatrix}}_{\dot{\boldsymbol{x}}(\mathrm{t})}$$
$$+ \underbrace{\begin{bmatrix} I_s(e^{\frac{v_1(t)-v_3(t)}{v_T}}-1) \\ 0 \\ -I_s(e^{\frac{v_1(t)-v_3(t)}{v_T}}-1) \\ 0 \end{bmatrix}}_{\boldsymbol{f}(\boldsymbol{x}(\mathrm{t}))} = \underbrace{\begin{bmatrix} I_{in}(t) \\ 0 \\ 0 \\ 0 \end{bmatrix}}_{\boldsymbol{b}(\mathrm{t})}, \tag{2.4}$$

which is now in a format that is suitable for efficient circuit simulation.

The remaining sections in this chapter review the main types of circuit analyses that are relevant to noise simulation. These simulation types are all detailed using the MNA formulation.

2.2 DC Analysis

DC analysis is a type of circuit analysis that mainly finds its applications in obtaining the solution to the system at the first time point for transient analysis and finding the operating point of the circuit for small-signal AC analysis. It involves finding the DC solution to the circuit due to the sole presence of constant sources.

With constant sources only, the system in (2.2) becomes:

$$\boldsymbol{G}\,\boldsymbol{x}(t) + \boldsymbol{C}\,\dot{\boldsymbol{x}}(t) + \boldsymbol{f}(\boldsymbol{x}(t)) = \boldsymbol{b}_{DC} \qquad \boldsymbol{x}(0) = \boldsymbol{x}_0, \tag{2.5}$$

where $\boldsymbol{b}_{DC}$ contains the contributions from the constant sources in the circuit and is now time-invariant. Furthermore, since $\boldsymbol{b}_{DC}$ is the input to the system

and is time-invariant, then the solution to the system $\boldsymbol{x}(t)$ is constant at equilibrium and its derivative $\dot{\boldsymbol{x}}(t)$ is equal to 0. As such, (2.5) at equilibrium can be simplified to

$$\boldsymbol{G}\,\boldsymbol{x}_{DC} + \boldsymbol{f}(\boldsymbol{x}_{DC}) = \boldsymbol{b}_{DC}, \tag{2.6}$$

where $\boldsymbol{x}_{DC}$ is the solution to the system at equilibrium. There are two ways to proceed from (2.6) depending on whether the circuit is linear or nonlinear.

2.2.1 DC Analysis for Linear Circuits

For the more specific case where there are no nonlinear components in the circuit, (2.6) simplifies to

$$\boldsymbol{G}\,\boldsymbol{x}_{DC} = \boldsymbol{b}_{DC}. \tag{2.7}$$

This simple system is then typically solved by lower-upper (LU) decomposition of the $\boldsymbol{G}$ matrix followed by forward and backward substitution, or by iterative solvers such as the generalized minimal residual method (GMRES) [29].

2.2.2 DC Analysis for Nonlinear Circuits

For the general case where there are some nonlinear components in the circuit, iterative numerical methods must be used to address the contributions from the nonlinear components. Among the most commonly used methods is the Newton-Raphson method, which attempts to find the roots of a real-valued function $\boldsymbol{\Psi}(\boldsymbol{x}_{DC})$. More precisely, the objective is to find $\boldsymbol{x}_{DC} \in R$ which best satisfies

$$R = \{\boldsymbol{r} \in \mathbb{R}^n \,\big|\, \boldsymbol{\Psi}(\boldsymbol{r}) = 0\} \tag{2.8}$$

for the desired level of accuracy. To do so, the real-valued function for the system in (2.6) is defined as

$$\boldsymbol{\Psi}(\boldsymbol{x}_{DC}) = \boldsymbol{G}\,\boldsymbol{x}_{DC} + \boldsymbol{f}(\boldsymbol{x}_{DC}) - \boldsymbol{b}_{DC}, \tag{2.9}$$

where $\boldsymbol{\Psi}(\boldsymbol{x}_{DC}) \in \mathbb{R}^n$ is equal to 0 when $\boldsymbol{x}_{DC}$ is the exact DC solution. Furthermore, its derivative is given by

$$\begin{aligned}\dot{\boldsymbol{\Psi}}\left(\boldsymbol{x}_{DC}^{(j)}\right) = \left.\frac{\partial \boldsymbol{\Psi}(\boldsymbol{x}_{DC})}{\partial \boldsymbol{x}_{DC}}\right|_{\boldsymbol{x}_{DC}=\boldsymbol{x}_{DC}^{(j)}} &= \boldsymbol{G} + \left.\frac{\partial \boldsymbol{f}(\boldsymbol{x}_{DC})}{\partial \boldsymbol{x}_{DC}}\right|_{\boldsymbol{x}_{DC}=\boldsymbol{x}_{DC}^{(j)}} \\ &= \boldsymbol{G} + \boldsymbol{J}\left(\boldsymbol{x}_{DC}^{(j)}\right),\end{aligned} \tag{2.10}$$

where $\boldsymbol{x}_{DC}^{(j)}$ is the current approximation of the exact value of $\boldsymbol{x}_{DC}$ at the j^{th} Newton-Raphson iteration and $\boldsymbol{J}\left(\boldsymbol{x}_{DC}^{(j)}\right)$ is the Jacobian matrix of $\boldsymbol{f}(\boldsymbol{x}_{DC})$ evaluated at $\boldsymbol{x}_{DC}^{(j)}$. Starting from an initial guess $\boldsymbol{x}_{DC}^{(0)}$, $\boldsymbol{x}_{DC}^{(j)}$ is updated iteratively as follows:

$$\boldsymbol{x}_{DC}^{(j+1)} = \boldsymbol{x}_{DC}^{(j)} + \Delta\boldsymbol{x}_{DC}^{(j)}, \tag{2.11}$$

where

$$\Delta\boldsymbol{x}_{DC}^{(j)} = -\left(\dot{\Psi}\left(\boldsymbol{x}_{DC}^{(j)}\right)\right)^{-1}\boldsymbol{\Psi}\left(\boldsymbol{x}_{DC}^{(j)}\right). \tag{2.12}$$

The desired level of accuracy for the convergence of the Newton-Raphson method can be implemented in two possible manners. The first is to set a user-defined error tolerance and determine if the algorithm converges on the j^{th} iteration based on whether or not the Euclidian norm of $\Delta\boldsymbol{x}_{DC}^{(j)}$ is below the error tolerance. The second is to set another user-defined error tolerance and verify if the algorithm converges on the j^{th} iteration based on whether or not the Euclidian norm of $\boldsymbol{\Psi}\left(\boldsymbol{x}_{DC}^{(j)}\right)$ is below the error tolerance. Moreover, it is possible to combine both of the above for a more accurate solution at the potential cost of increasing the number of iterations required for the algorithm to converge. Finally, a careful selection of the initial guess $\boldsymbol{x}_{DC}^{(0)}$, $\boldsymbol{x}_{DC}^{(j)}$ such that it is closer to the exact solution results in faster convergence and can often determine whether the algorithm will converge at all [21].

Continuation Methods for Convergence Problems

As nonlinear DC analysis is an important tool for various types of noise analysis methods, it is important to discuss its shortcomings. Should iterative techniques such as Newton-Raphson fail to converge using an arbitrarily selected initial guess $\boldsymbol{x}_{DC}^{(0)}$, continuation methods can be used [6, 27]. When applied to the formulation in (2.6), the nonlinear system becomes:

$$\boldsymbol{G}\,\boldsymbol{x}_{DC} + \boldsymbol{f}(\boldsymbol{x}_{DC}) = \alpha\,\boldsymbol{b}_{DC}, \tag{2.13}$$

where α is an extra parameter. The input source vector $\boldsymbol{b}_{DC}$ is then incrementally increased starting from zero using an increasing sequence of values for $\alpha \in [0, 1]$. In doing so, (2.13) can initially be solved in a trivial manner and provide progressively better initial guesses to solve the subsequent problem which is closer to the original problem. Naturally, a larger sequence of values for α increases the likelihood that the iterative technique will converge at the expense of a larger CPU cost.

2.3 Transient Time Domain Analysis

Transient analysis is a simulation type that observes the changes that occur in a circuit over time. It is often used to simulate the transient part of the circuit response before steady state is reached. Starting from a given initial condition, performing a transient simulation involves computing the solution to the system at every time point using the solution at previous time points. The resulting transient response can be interpreted as a trajectory in time, the transient part of which is shaped by the initial condition. Numerical integration methods are used to solve systems of ordinary differential equations (ODEs) such as (2.2) in the time domain. More specifically, linear multistep methods are used to approximate the time derivative $\dot{\boldsymbol{x}}(t)$ at every time point [4].

Let be t_k be a discrete time point, with $0 \leq \ldots < t_k < t_{k+1} < \ldots$ and $k = 0, 1, 2, \ldots$. By discretizing it in time and writing it at time $t = t_{k+1}$, (2.2) becomes:

$$\boldsymbol{G}\,\boldsymbol{x}_{k+1} + \boldsymbol{C}\,\dot{\boldsymbol{x}}_{k+1} + \boldsymbol{f}(\boldsymbol{x}_{k+1}) = \boldsymbol{b}_{k+1}, \tag{2.14}$$

where $\boldsymbol{x}_{k+1}$ is defined as $\boldsymbol{x}(t_{k+1})$ for simplicity and $\boldsymbol{x}_0$ is equal to the DC solution to the circuit (see Section 2.2). The backward Euler method detailed in Table 2–1 is used to solve (2.14) as it will be the integration method of choice in this thesis. Using the backward Euler method to approximate the derivative in (2.14) results in the following difference equation:

$$\boldsymbol{G}\,\boldsymbol{x}_{k+1} + \boldsymbol{C}\left(\frac{\boldsymbol{x}_{k+1} - \boldsymbol{x}_k}{h}\right) + \boldsymbol{f}(\boldsymbol{x}_{k+1}) = \boldsymbol{b}_{k+1}, \tag{2.15}$$

where h is the time step that separates two successive time points. Rearranging (2.15) such that all the known terms are located on the right-hand side of the equation yields the following difference equation:

$$\left(\boldsymbol{G} + \frac{\boldsymbol{C}}{h}\right)\boldsymbol{x}_{k+1} + \boldsymbol{f}(\boldsymbol{x}_{k+1}) = \frac{\boldsymbol{C}}{h}\boldsymbol{x}_k + \boldsymbol{b}_{k+1}. \tag{2.16}$$

The process of solving for $\boldsymbol{x}_{k+1}$ in (2.16) varies depending on whether the circuit is linear or nonlinear.

Table 2–1: Formulae for the numerical approximation of time derivatives for common linear multistep methods.

Linear Multistep Method	**Numerical Approximation**
Forward Euler	$\dot{\boldsymbol{x}}(t_k) = \frac{\boldsymbol{x}(t_{k+1}) - \boldsymbol{x}(t_k)}{h}$
Backward Euler	$\dot{\boldsymbol{x}}(t_{k+1}) = \frac{\boldsymbol{x}(t_{k+1}) - \boldsymbol{x}(t_k)}{h}$
Trapezoidal Rule	$\frac{\dot{\boldsymbol{x}}(t_{k+1}) + \dot{\boldsymbol{x}}(t_k)}{2} = \frac{\boldsymbol{x}(t_{k+1}) - \boldsymbol{x}(t_k)}{h}$

2.3.1 Backward Euler Method for Linear Circuits

For the more specific case where there are no nonlinear components in the circuit, (2.16) simplifies to

$$\left(\boldsymbol{G}+\frac{\boldsymbol{C}}{h}\right)\boldsymbol{x}_{k+1}=\frac{\boldsymbol{C}}{h}\boldsymbol{x}_k+\boldsymbol{b}_{k+1}. \tag{2.17}$$

Since $\left(\boldsymbol{G}+\frac{\boldsymbol{C}}{h}\right)$ is a constant matrix and the right-hand side of (2.17) is known, the main CPU cost of the transient simulation of a linear circuit is a single LU decomposition of $\left(\boldsymbol{G}+\frac{\boldsymbol{C}}{h}\right)$ for the entire simulation combined with one forward substitution and backward substitution at every time point.

2.3.2 Backward Euler Method for Nonlinear Circuits

For the general case where there are some nonlinear components in the circuit, (2.16) can be solved using a nonlinear iterative solver such as Newton-Raphson. To do so, the real-valued function for the system in (2.16) is defined as

$$\boldsymbol{\Psi}(\boldsymbol{x}_{k+1})=\boldsymbol{G}\,\boldsymbol{x}_{k+1}+\boldsymbol{C}\left(\frac{\boldsymbol{x}_{k+1}-\boldsymbol{x}_k}{h}\right)+\boldsymbol{f}(\boldsymbol{x}_{k+1})-\boldsymbol{b}_{k+1}. \tag{2.18}$$

Furthermore, its derivative is given by

$$\begin{aligned}\dot{\boldsymbol{\Psi}}\left(\boldsymbol{x}_{k+1}^{(j)}\right)=\left.\frac{\partial\boldsymbol{\Psi}(\boldsymbol{x}_{k+1})}{\partial\boldsymbol{x}_{k+1}}\right|_{\boldsymbol{x}_{k+1}=\boldsymbol{x}_{k+1}^{(j)}}&=\left(\boldsymbol{G}+\frac{\boldsymbol{C}}{h}\right)+\left.\frac{\partial\boldsymbol{f}(\boldsymbol{x}_{k+1})}{\partial\boldsymbol{x}_{k+1}}\right|_{\boldsymbol{x}_{k+1}=\boldsymbol{x}_{k+1}^{(j)}}\\&=\left(\boldsymbol{G}+\frac{\boldsymbol{C}}{h}\right)+\boldsymbol{J}\left(\boldsymbol{x}_{k+1}^{(j)}\right),\end{aligned} \tag{2.19}$$

where $\boldsymbol{x}_{k+1}^{(j)}$ is the current approximation of the exact value of $\boldsymbol{x}_{k+1}$ at the j^{th} Newton-Raphson iteration and $\boldsymbol{J}\left(\boldsymbol{x}_{k+1}^{(j)}\right)$ is the Jacobian matrix of $\boldsymbol{f}(\boldsymbol{x}_{k+1})$ evaluated at $\boldsymbol{x}_{k+1}^{(j)}$. Starting from $\boldsymbol{x}_0$, $\boldsymbol{x}_{k+1}^{(j)}$ is updated iteratively as follows:

$$\boldsymbol{x}_{k+1}^{(j+1)}=\boldsymbol{x}_{k+1}^{(j)}-\left(\dot{\Psi}\left(\boldsymbol{x}_{k+1}^{(j)}\right)\right)^{-1}\boldsymbol{\Psi}\left(\boldsymbol{x}_{k+1}^{(j)}\right). \tag{2.20}$$

Contrary to the DC analysis of a nonlinear circuit for which the Newton-Raphson method only needs to be applied once, the transient analysis of a nonlinear

circuit requires the application of the Newton-Raphson method at every time point. The main CPU cost can therefore be attributed to the figurative inversion of $\dot{\Psi}\left(\boldsymbol{x}_{k+1}^{(j)}\right)$ for every Newton-Raphson iteration at each time point. Moreover, as with the DC analysis of a nonlinear circuit, convergence problems may arise. In such a case, the continuation method described in subsection 2.2.2 can be applied in the same way to ensure convergence.

2.4 Linear Frequency Domain Analysis

Linear frequency domain analysis is a simulation type that provides the steady state response of a circuit with respect to periodic inputs over a certain frequency range and is hence commonly known as AC analysis. It can be applied to linear circuits or nonlinear circuits that are linearized around an operating point (i.e. small-signal analysis) using phasor analysis [25]. Small-signal equivalent circuits for many integrated components have been established [12].

2.4.1 AC Analysis for Linear Circuits

For the AC analysis of a linear circuit, the MNA formulation in (2.2) features no contributions from $\boldsymbol{f}(\boldsymbol{x}(t))$ and the initial condition is omitted as it does not affect the steady state response of the circuit. Consequently, the formulation becomes:

$$\boldsymbol{G}\,\boldsymbol{x}(t) + \boldsymbol{C}\,\dot{\boldsymbol{x}}(t) = \boldsymbol{b}(t). \tag{2.21}$$

Next, consider an arbitrary sinusoidal input signal operating at a frequency $\omega = 2\pi f$ whose contribution is stored in $\boldsymbol{b}(t)$. The phasor form of $\boldsymbol{b}(t)$ can then be expressed as:

$$\boldsymbol{b}(t) \Rightarrow \boldsymbol{B}e^{j\omega t}, \tag{2.22}$$

where $\boldsymbol{B} \in \mathbb{C}^n$ is a vector which contains information on the magnitude and phase of $\boldsymbol{b}(t)$. Because the circuit is linear, the steady state response of the circuit is also sinusoidal with the same frequency ω. Consequently, the solution

vector $\boldsymbol{x}(t)$ can be expressed in phasor form as:

$$\boldsymbol{x}(t) = \boldsymbol{X} e^{j\omega t}. \tag{2.23}$$

Substituting (2.22) and (2.23) into (2.21) results in the following equation:

$$\boldsymbol{G}\,\boldsymbol{X} e^{j\omega t} + \boldsymbol{C}\,\boldsymbol{X} \frac{\partial e^{j\omega t}}{\partial t} = \boldsymbol{B} e^{j\omega t}, \tag{2.24}$$

which can be further simplified to:

$$\boldsymbol{G}\,\boldsymbol{X} + j\omega \boldsymbol{C}\,\boldsymbol{X} = \boldsymbol{B}. \tag{2.25}$$

Solving for $\boldsymbol{X}$ in (2.25) leads to the following equation:

$$\boldsymbol{X} = (\boldsymbol{G} + j\omega \boldsymbol{C})^{-1}\,\boldsymbol{B}. \tag{2.26}$$

Typically, a sweep over a given frequency range is performed and the value of ω changes at every frequency point. Hence, $\boldsymbol{X}$ contains not only the frequency domain magnitude of the circuit response but also its phase. Furthermore, the main CPU cost of performing a linear frequency domain simulation lies in the figurative matrix inversion at every frequency point.

2.4.2 AC Analysis for Nonlinear Circuits

For the AC analysis of a nonlinear circuit operating under small-signal conditions, the solution vector $\boldsymbol{x}(t)$ can be divided into two parts as follows:

$$\boldsymbol{x}(t) = \boldsymbol{x}_{DC} + \boldsymbol{x}_s(t), \tag{2.27}$$

where $\boldsymbol{x}_{DC}$ is the solution to the system in (2.6) due to constant inputs (i.e. the DC solution) and $\boldsymbol{x}_s(t)$ denotes the small-signal solution to the circuit due to small periodic input signals. Noting that $\dot{\boldsymbol{x}}_{DC} = 0$, the MNA formulation in (2.2) can then be expanded like so:

$$\boldsymbol{G}(\boldsymbol{x}_{DC} + \boldsymbol{x}_s(t)) + \boldsymbol{C}\,\dot{\boldsymbol{x}}_s(t) + \boldsymbol{f}(\boldsymbol{x}_{DC} + \boldsymbol{x}_s(t)) = \boldsymbol{b}_{DC} + \boldsymbol{b}_s(t), \tag{2.28}$$

where $\boldsymbol{b}_{DC}$ and $\boldsymbol{b}_s(t)$ respectively contain the contributions from the constant input sources and small periodic input signals. The first-order Taylor approximation of $\boldsymbol{f}(\boldsymbol{x}(t))$ around $\boldsymbol{x}_{DC}$ is given by

$$\boldsymbol{f}(\boldsymbol{x}(t)) \cong \boldsymbol{f}(\boldsymbol{x}_{DC}) + \left.\frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}(t)=\boldsymbol{x}_{DC}} (\boldsymbol{x}(t) - \boldsymbol{x}_{DC}). \tag{2.29}$$

To linearize the system in (2.28), (2.29) is used to approximate $\boldsymbol{f}(\boldsymbol{x}_{DC}+\boldsymbol{x}_s(t))$ as follows:

$$\begin{aligned} \boldsymbol{f}(\boldsymbol{x}_{DC} + \boldsymbol{x}_s(t)) &\cong \boldsymbol{f}(\boldsymbol{x}_{DC}) + \left.\frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}(t)=\boldsymbol{x}_{DC}} (\boldsymbol{x}_{DC} + \boldsymbol{x}_s(t) - \boldsymbol{x}_{DC}) \\ &= \boldsymbol{f}(\boldsymbol{x}_{DC}) + \boldsymbol{J}(\boldsymbol{x}_{DC})\, \boldsymbol{x}_s(t), \end{aligned} \tag{2.30}$$

where $\boldsymbol{J}(\boldsymbol{x}_{DC})$ is the Jacobian matrix of $\boldsymbol{x}(t)$ evaluated at $\boldsymbol{x}_{DC}$. It is important to note that this approximation is only valid for circuits operating under small-signal conditions.

Substituting (2.30) in (2.28) and subtracting (2.6) from (2.28) yields the following equation:

$$(\boldsymbol{G} + \boldsymbol{J}(\boldsymbol{x}_{DC}))\, \boldsymbol{x}_s(t) + \boldsymbol{C}\, \dot{\boldsymbol{x}}_s(t) \cong \boldsymbol{b}_s(t). \tag{2.31}$$

The formulation in (2.31) strongly resembles the one in (2.21), with the main difference being that the coefficient matrix of $\boldsymbol{x}(t)$ in the former is $\boldsymbol{G}+\boldsymbol{J}(\boldsymbol{x}_{DC})$ rather than simply $\boldsymbol{G}$, where $\boldsymbol{J}(\boldsymbol{x}_{DC})$ is a constant. This implies that once the circuit has been linearized around $\boldsymbol{x}_{DC}$ and hence $\boldsymbol{J}(\boldsymbol{x}_{DC})$ has been obtained, the process of performing the remainder of the AC analysis reduces to the process of performing the AC analysis of a linear circuit.

CHAPTER 3
Noise Analysis

3.1 Introduction

In this chapter, a detailed description of the methodology for the noise analysis of an electronic system is provided. The word *noise* is used in this thesis to refer to the electrical noise that is generated by the circuit components themselves. The term *noise analysis* then denotes the prediction of the small current and voltage fluctuations incurred by the presence of noise sources in a circuit. As there are innumerable unique circuits in existence that exhibit distinct behaviors, it follows that there are many different noise analysis methods as some are better suited for a specific type of circuit than others. Noise analysis methods are classified in this chapter into frequency domain methods and time domain methods.

The methodology for noise analysis typically comprises the following four components:

- Mathematical models for the noise sources;
- Mathematical formulation of the noisy system;
- A noise analysis method to simulate the effects of noise on the circuit;
- Techniques to calculate the noise characteristics using the data from the noise simulation.

3.2 Noise Models

To model the fluctuations in the amplitude and phase of the signals in a circuit due to a noisy component, the noisy component can be replaced by either a noise voltage source in series with the noiseless model of the noisy

component or a noise current source in parallel with the noiseless model of the noisy component as shown in Fig. 3–1.

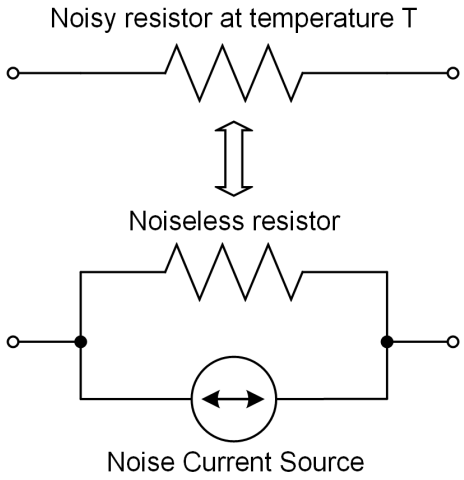

Figure 3–1: Thermal noise model for a noisy resistor.

Due to the random nature of noise, the exact value of these noise sources is unknown. However, what is known is the power spectral density (PSD) of the noise sources, which varies according to the characteristics of the noisy component as well as the type(s) of noise produced by the noisy component. There are three main types of noise: thermal noise, shot noise, and flicker noise.

3.2.1 Thermal Noise

Thermal noise occurs in any resistive material and is caused by the random motion of electrons due to thermal agitation. At room temperature, the bandwidth of thermal noise is approximately 6000 GHz, and time samples of thermal noise are thus uncorrelated when separated by more than 0.17 ps [10]. For frequencies below the higher end of this bandwidth, the PSD of thermal noise is constant with frequency. As such, thermal noise is modeled as white noise. The double-sided PSD of thermal noise, or current variance

(mean square) per Hertz of bandwidth, is given by

$$S_{th}(t) = 2kTG(t), \tag{3.1}$$

where k is the Boltzmann constant, T is the temperature in K, and $G(t) \in \mathbb{R}$ is the time-varying conductance in Ω^{-1}.

3.2.2 Shot Noise

Shot noise stems from the discrete nature of charge carriers and is more prevalent in devices that have a p-n junction, such as diodes and bipolar transistors. The lifetime of a shot event is typically fractions of a nanosecond for semiconductor devices, so the bandwidth of shot noise is usually in the high gigahertz range [10]. As with thermal noise, the PSD of shot noise is constant with frequency below the high end of the gigahertz region and shot noise is thus also modeled as white noise. The double-sided PSD of shot noise, or current variance (mean square) per Hertz of bandwidth, is given by

$$S_{sh}(t) = qI_D(t), \tag{3.2}$$

where q is the electron charge in C and $I_D(t)$ is the time-varying current through the junction in A.

3.2.3 Flicker Noise

Flicker noise is present in all active elements as well as some passive elements and is caused by a variety of phenomena. Unlike thermal and shot noise, the PSD of flicker noise decreases with frequency. Consequently, flicker noise is modeled as pink noise rather than white noise and it is a low-frequency phenomenon due to how white noise from other sources dominates it at higher frequencies. The double-sided PSD of flicker noise is given by

$$S_{fl}(f) = K\frac{I^{\alpha}}{f^{\beta}}, \tag{3.3}$$

where K, α, and β are constants particular to each device, and I is the current through the device in A.

Unlike thermal and shot noise, flicker noise does not have independent values at every time point and is correlated with both recent values and values in the distant past [20]. To include them in a unified formulation for time domain noise analysis later on in this chapter, flicker noise sources need to be synthesized using white noise sources. One way to do so is to use the summation of Lorentzian spectra to model the PSD of each individual flicker noise source [14]. In a time domain noise simulation, the summation of Lorentzian spectra that models the PSD of a given flicker noise source can be obtained by replacing the flicker noise source with a series combination of noisy resistors in parallel with a capacitor. The thermal noise that is generated by these resistor-capacitor blocks is thus filtered in a way that produces the desired summation of Lorentzian spectra that models the PSD of the flicker noise source at the output of this flicker noise synthesis circuit. A summation of N Lorentzian spectra is given by

$$S(f) = \frac{2\sigma^2}{\pi} \sum_{h=1}^{N} \frac{p_h}{p_h^2 + f^2}, \tag{3.4}$$

where p_h designates the pole frequencies and σ^2 is the variance of the considered quantity [26]. In the context of a flicker noise synthesis circuit being used to generate such a PSD, the pole frequencies correspond to the poles introduced by the resistor-capacitor blocks and the variance becomes that of the so-called kTC noise that is produced by resistor-capacitor circuits.

Naturally, the size of the overall circuit for the purpose of time domain noise simulation and hence the complexity of the noise simulation itself increases as a consequence of modeling flicker noise sources using flicker noise synthesis circuits. The number of extra nodes introduced by modeling flicker noise

sources in the time domain in this manner is directly related to their β constants in (3.3) as well as the frequency range over which their PSDs are modeled in the frequency domain [26].

3.3 Noise Bandwidth

The noise bandwidth used for simulation purposes, which is denoted by F_{max}, is a concept that is relevant to both frequency domain and time domain noise analysis. As seen previously, the different noise types have their own PSDs which eventually fall off as the frequency increases. In the case of thermal and shot noise, the PSD is constant and only starts the decrease at tremendously high frequencies. Thermal and shot noise can therefore be considered as having their own bandwidths. Oftentimes, the bandwidth of a circuit is much smaller than the bandwidth of the noise sources it contains.

At first glance, it would seem sufficient to choose a slightly bigger F_{max} than the circuit bandwidth to fully capture the output noise PSD. As will be explained in the next two sections, F_{max} has a direct effect on the overall noise simulation time for frequency domain noise analysis methods as well as some time domain noise analysis methods, so there is often an incentive to limit F_{max} to decrease computation time. However, because the output noise PSD is calculated from the individual contributions from all the noise sources in the circuit that act as their own voltage or current sources, the circuit bandwidth for noise analysis is better defined as various individual bandwidths from the noise sources to the output rather than as a single bandwidth between the input signal and the output. As such, noise sources located at different locations in a circuit will also have different bandwidths with respect to the output depending on what filtering effects exist between the noise sources and the output. It then becomes important to properly model the parasitic capacitances which exist within the circuit as they have a significant impact

on the bandwidth of some noise sources. For example, the thermal noise in the node voltage $v_2(t)$ in Fig. 2–1 caused by the resistor R_2 will have a very high value in the noise simulation as it is effectively unfiltered from the point of view of the noise simulation. This is because there is no capacitive path to ground from that node, only an inductive path. A capacitive path to ground would progressively start behaving like a short circuit for the noise current to go through as the frequency is increased to high values, whereas the inductor progressively starts acting as an open circuit that blocks access to ground instead. In practice, parasitic capacitances in a circuit have a filtering effect on the noise, so modeling them adequately to avoid unfiltered noise at some nodes is necessary for accurate noise simulations.

Furthermore, it is often the case that a few noise sources in the circuit have a very large bandwidth compared to the other noise sources. For instance, a noise source that is near the output and for which the only filtering effect is caused by a capacitive path to ground through a parasitic capacitor will have a very large bandwidth with respect to the output as well as a significant impact on the total output noise. Therefore, to accurately simulate the total output noise, it is often necessary to pick a much larger F_{max} than the traditionally defined circuit bandwidth to fully capture high-frequency noise from all the noise sources. This gives rise to a trade-off between noise simulation accuracy and overall noise simulation time through careful selection of F_{max} for most noise simulation methods. For accurate noise simulations, however, F_{max} should be slightly higher than the largest individual bandwidth(s) in the circuit to avoid neglecting oftentimes significant contributions from some noise sources, but not so high as to exceed the noise bandwidths discussed in Section 3.2. A visual representation of the selection of F_{max} for accurate noise simulations is given in Fig. 3–2.

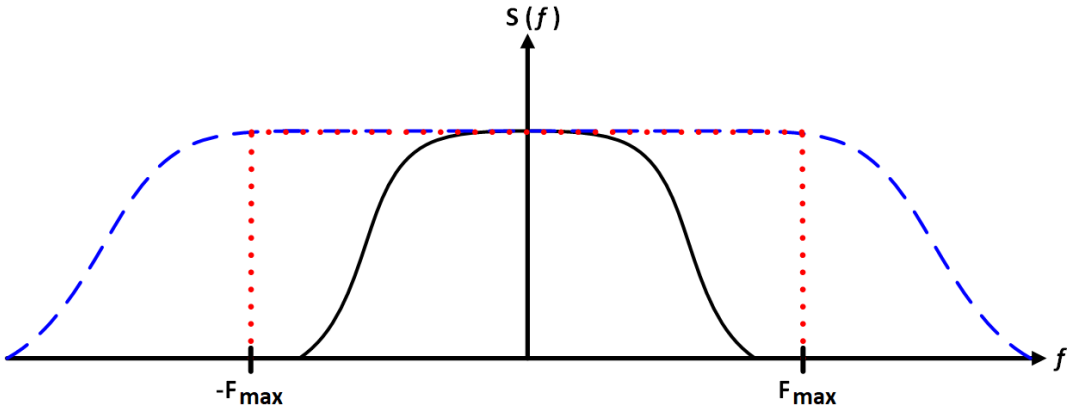

Figure 3–2: Actual output noise PSD (solid line), theoretical noise PSD (dashed line), and noise PSD used for simulation purposes (dotted line).

3.4 Frequency Domain Noise Analysis

Noise analysis in the frequency domain is typically conducted in a small-signal fashion. Small-signal equivalent circuits for many integrated components have been built around noise simulation [12]. The main assumption associated with small-signal noise analysis is that the noise in the circuit is much smaller than the signal and does not significantly affect the operating point or periodic state of interest. Consequently, frequency domain noise analysis becomes a linear problem that stems from the linearization of the circuit around its operating point or periodic state.

When performing a small-signal noise simulation, the individual contributions from every noise source in the circuit to the output are computed and then compounded. All the noise sources in the circuit are assumed to be independent Gaussian processes with a known PSD and random phase. The noise sources thus provide their own contributions to the total output noise, where each contribution is given by

$$\sigma^2 = \int_{-\infty}^{\infty} S_o(f)df, \tag{3.5}$$

where σ^2 is the variance of the output noise due to a given noise source in V^2 and $S_o(f)$ is the double-sided PSD of the output noise incurred by that noise

source in V^2/Hz. Once an appropriate value for F_{max} has been selected, (3.5) simplifies to

$$\sigma^2 = \int_{-F_{max}}^{F_{max}} S_o(f)df. \tag{3.6}$$

Then, as the noise sources are assumed to be uncorrelated (independent Gaussian processes are uncorrelated by definition), the total output noise is given by

$$\sigma^2_{total} = \sum_i \sigma_i^2, \tag{3.7}$$

where σ^2_{total} is the total output noise variance in V^2.

The bulk of the computations for frequency domain noise simulation lies in computing the PSD of the output noise incurred by every single noise source in the circuit as it involves separately calculating the transfer function between the noise sources and the output at every frequency point. As such, this process can become computationally expensive when F_{max} is large and the number of noise sources in the circuit is high. One way to make frequency domain noise analysis more efficient is to widen the interval between subsequent frequency points in the frequency regions which see little change in the frequency response. Typically, this translates to simulating progressively fewer frequency points as the frequency increases.

Moreover, there are two similar ways of computing the individual contributions from the noise sources to the output PSD of a circuit depending on whether the circuit of interest is a linear time-invariant (LTI) circuit or linear periodically time-varying (LPTV) circuit. More precisely, the nature of the circuit determines the relationship between the PSD of the noise sources and that of the output noise.

3.4.1 Noise Analysis for LTI Circuits

Noise analysis for LTI circuits is straightforward. The nonlinear circuit is linearized around a fixed operating point, which provides a LTI system

for noise analysis. The PSD of the output noise incurred by the individual contribution of a noise source is then given by

$$S_o(f) = \left|H(f)\right|^2 S_i(f), \tag{3.8}$$

where $S_i(f)$ is the PSD of the noise source in A^2/Hz, $S_o(f)$ is the PSD of the output noise incurred by that noise source in V^2/Hz, and $H(f)$ is the transfer function between the noise source and the output in $\mathrm{V\,A}^{-1}$. The underlying assumption behind (3.8) is that both the noise sources as well as the output noise are WSS stochastic processes. However, this assumption is only valid for LTI circuits.

3.4.2 Noise Analysis for LPTV Circuits

For circuits with changing bias conditions or circuits for which small-signal conditions may not be assumed, a generalization of the noise analysis method for LTI circuits is required. One way to do so is to linearize the circuit around a periodic steady-state rather than a fixed operating point, which provides a LPTV system rather than a LTI system for noise analysis [16]. The average PSD of the output noise incurred by the individual contribution of a noise source is then given by

$$\overline{S}_o(f) = \sum_{n=-\infty}^{\infty} \left|H_n(f)\right|^2 S_i\left(f + n\frac{1}{T}\right), \tag{3.9}$$

where $S_i(f)$ is the PSD of the noise source in A^2/Hz, $\overline{S}_o(f)$ is the average PSD of the output noise incurred by that noise source in V^2/Hz, $H_n(f)$ represents the Fourier coefficients of the transfer function between the noise source and the output in $\mathrm{V\,A}^{-1}$, and T is the period. The formulation in (3.9) relies on the assumptions that the circuit features periodic large-signal excitations and periodic steady-state large-signal waveforms and hence the noise sources as well as the output are cyclostationary stochastic processes. In fact, the

noise sources can be modeled as WSS stochastic processes by exploiting the assumption that they are cyclostationary noise sources [15]. However, this noise analysis method is only applicable to LPTV circuits in which the noise can be assumed to be cyclostationary.

3.5 Time Domain Noise Analysis

Noise analysis in the time domain is typically conducted through transient analysis. Transient noise analysis is used to simulate the evolution of the noise in a circuit over time as opposed to frequency domain noise analysis in which the noise is simulated at a fixed point in time. This gives rise to completely different kinds of noise analysis methods, which are categorized in this chapter as Monte Carlo and non-Monte Carlo methods. As they all rely on transient analysis to simulation noise, these methods share a common formulation of noisy circuit equations.

3.5.1 System Formulation

Based on the formulation in (2.1), a general circuit with noisy components can be modeled as the following system of stochastic differential equations (SDEs):

$$\boldsymbol{F}(\dot{\boldsymbol{x}}(t), \boldsymbol{x}(t), t) + \boldsymbol{B}(\boldsymbol{x}(t), t)\, \boldsymbol{v} = 0 \qquad \boldsymbol{x}(0) = \boldsymbol{x}_0 + \boldsymbol{x}_{noise,0}, \tag{3.10}$$

where $\boldsymbol{B}\left(\boldsymbol{x}(t), t\right) \in \mathbb{R}^{n \times q}$ is a matrix containing the noise currents which model the q total number of noise sources, $\boldsymbol{v}$ is a vector of q standard white Gaussian stochastic processes, and $\boldsymbol{x}_{noise,0}$ is a vector of n random variables with zero mean. It should be noted that $\boldsymbol{v}$ is implicitly a function of t in that it is updated with the simulation time step Δt. Every column in $\boldsymbol{B}(\boldsymbol{x}(t), t)$ corresponds to a single noise current source that can have either one or two nonzero entries depending on whether one of the two nodes between which the noise source is located is grounded or not.

When implemented using the MNA formulation in (2.2), (3.10) becomes:

$$\boldsymbol{G}\,\boldsymbol{x}(t)+\boldsymbol{C}\,\dot{\boldsymbol{x}}(t)+\boldsymbol{f}(\boldsymbol{x}(t))+\boldsymbol{B}(\boldsymbol{x}(t),t)\,\boldsymbol{v}=\boldsymbol{b}(t) \qquad \boldsymbol{x}(0)=\boldsymbol{x}_0+\boldsymbol{x}_{noise,0}. \tag{3.11}$$

Let $\boldsymbol{x}_{sn}(t)$ be the solution to the noisy system in (3.11). The system then becomes:

$$\begin{aligned}\boldsymbol{G}\,\boldsymbol{x}_{sn}(t)+\boldsymbol{C}\,\dot{\boldsymbol{x}}_{sn}(t)+\boldsymbol{f}(\boldsymbol{x}_{sn}(t))+\boldsymbol{B}(\boldsymbol{x}_{sn}(t),t)\,\boldsymbol{v}&=\boldsymbol{b}(t)\\ \boldsymbol{x}_{sn}(0)&=\boldsymbol{x}_0+\boldsymbol{x}_{noise,0}.\end{aligned} \tag{3.12}$$

Accordingly, the noise in the solution is defined as

$$\boldsymbol{x}_{noise}=\boldsymbol{x}_{sn}-\boldsymbol{x}_s, \tag{3.13}$$

where $\boldsymbol{x}_s$ is the solution to the system without noise. It should be noted that the time dependency of $\boldsymbol{x}_{sn}$, $\boldsymbol{x}_s$, and $\boldsymbol{x}_{noise}$ is henceforth omitted for simplicity. The first-order Taylor approximation of $\boldsymbol{f}(\boldsymbol{x})$ around $\boldsymbol{x}_s$ is given by

$$\boldsymbol{f}(\boldsymbol{x})\cong\boldsymbol{f}(\boldsymbol{x}_s)+\left.\frac{\partial\boldsymbol{f}(\boldsymbol{x})}{\partial\boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{x}_s}(\boldsymbol{x}-\boldsymbol{x}_s)=\boldsymbol{f}(\boldsymbol{x}_s)+\boldsymbol{J}(\boldsymbol{x}_s)(\boldsymbol{x}-\boldsymbol{x}_s), \tag{3.14}$$

where $\boldsymbol{J}(\boldsymbol{x}_s)$ is the Jacobian matrix of $\boldsymbol{f}(\boldsymbol{x})$ evaluated at $\boldsymbol{x}_s$. Considering that the noise in a signal is typically many orders of magnitude smaller than the signal itself (i.e. $\boldsymbol{x}_{noise}\ll\boldsymbol{x}_s$), (3.14) can be used to approximate $\boldsymbol{f}(\boldsymbol{x}_{sn})$ in (3.12) as follows:

$$\begin{aligned}\boldsymbol{G}\,\boldsymbol{x}_{sn}+\boldsymbol{C}\,\dot{\boldsymbol{x}}_{sn}+\boldsymbol{f}(\boldsymbol{x}_s)+\boldsymbol{J}(\boldsymbol{x}_s)(\boldsymbol{x}_{sn}-\boldsymbol{x}_s)+\boldsymbol{B}(\boldsymbol{x}_{sn},t)\,\boldsymbol{v}&\cong\boldsymbol{b}(t)\\ \boldsymbol{x}_{sn}(0)&=\boldsymbol{x}_0+\boldsymbol{x}_{noise,0}.\end{aligned} \tag{3.15}$$

The noisy system is now a system of *linear* SDEs due to the small noise perturbation approximation in (3.14), which allows the time domain noise analysis methods detailed in the later sections to be applied to a linear system.

Furthermore, $\boldsymbol{B}(\boldsymbol{x}_{sn}, t)$ can be approximated as

$$\boldsymbol{B}(\boldsymbol{x}_{sn}, t) \cong \boldsymbol{B}(\boldsymbol{x}_s, t) \tag{3.16}$$

and then defined as

$$\boldsymbol{B}(t) = \boldsymbol{B}(\boldsymbol{x}_s, t) \tag{3.17}$$

for simplicity due to the previous assumption that the noise in a signal is much smaller than the signal itself. In fact, the approximation in (3.16) allows for the generation of $\boldsymbol{B}(t)$ from a transient noiseless simulation prior to the noise simulation itself as it now only depends on $\boldsymbol{x}_s$. One of the key concepts of transient noise analysis is that the mean μ and the variance σ^2 of $\boldsymbol{x}_{sn}$ are given by

$$\mu(\boldsymbol{x}_{sn}) = \boldsymbol{x}_s + \mu(\boldsymbol{x}_{noise}) \tag{3.18}$$

and

$$\sigma^2(\boldsymbol{x}_{sn}) = \sigma^2(\boldsymbol{x}_{noise}) \tag{3.19}$$

due to the fact that $\boldsymbol{x}_s$ in (3.13) is deterministic by definition. It follows that a transient noise simulation with respect to $\boldsymbol{x}_{noise}$ provides the same information about the noise in a circuit as a transient noise simulation with respect to $\boldsymbol{x}_{sn}$ would. With this idea in mind, the noisy system in (3.15) can be restructured around $\boldsymbol{x}_{noise}$ in the following way. First, consider the noiseless system in (2.2) to which the solution is $\boldsymbol{x}_s$:

$$\boldsymbol{G}\,\boldsymbol{x}_s + \boldsymbol{C}\,\dot{\boldsymbol{x}}_s + \boldsymbol{f}(\boldsymbol{x}_s) = \boldsymbol{b}(t) \qquad \boldsymbol{x}_s(0) = \boldsymbol{x}_0. \tag{3.20}$$

Next, substituting (3.16) and (3.17) in (3.15) and expanding $\boldsymbol{x}_{sn}$ using (3.13) results in the following:

$$\begin{aligned} \boldsymbol{G}(\boldsymbol{x}_s + \boldsymbol{x}_{noise}) + \boldsymbol{C}(\dot{\boldsymbol{x}}_s + \dot{\boldsymbol{x}}_{noise}) + \boldsymbol{f}(\boldsymbol{x}_s) \\ + \boldsymbol{J}(\boldsymbol{x}_s)\boldsymbol{x}_{noise} + \boldsymbol{B}(t)\,\boldsymbol{v} \cong \boldsymbol{b}(t) \\ \boldsymbol{x}_s(0) + \boldsymbol{x}_{noise}(0) = \boldsymbol{x}_0 + \boldsymbol{x}_{noise,0}. \end{aligned} \tag{3.21}$$

Then, to remove all the terms that do not have an effect on $\boldsymbol{x}_{noise}$ and its derivative, (3.20) is subtracted from (3.21) resulting in the following noise system:

$$(\boldsymbol{G} + \boldsymbol{J}(\boldsymbol{x}_s))\,\boldsymbol{x}_{noise} + \boldsymbol{C}\,\dot{\boldsymbol{x}}_{noise} + \boldsymbol{B}(t)\,\boldsymbol{v} \cong 0 \qquad \boldsymbol{x}_{noise}(0) = \boldsymbol{x}_{noise,0}. \tag{3.22}$$

Finally, define:

$$\boldsymbol{A}(t) = \boldsymbol{G} + \boldsymbol{J}(\boldsymbol{x}_s), \tag{3.23}$$

where $\boldsymbol{A}(t)$ is time-varying for nonlinear circuits and time-invariant and simply equal to the $\boldsymbol{G}$ matrix when the circuit is linear. The noise system in (3.22) can then be simplified to

$$\boldsymbol{A}(t)\,\boldsymbol{x}_{noise} + \boldsymbol{C}\,\dot{\boldsymbol{x}}_{noise} + \boldsymbol{B}(t)\,\boldsymbol{v} \cong 0 \qquad \boldsymbol{x}_{noise}(0) = \boldsymbol{x}_{noise,0}, \tag{3.24}$$

which will be the basis for the two transient noise analysis methods detailed in this section.

It should be noted that $\boldsymbol{b}(t)$ no longer has an explicit effect on the system to solve. However, $\boldsymbol{b}(t)$ implicitly affects the linearization process (i.e. the computation of $\boldsymbol{J}(\boldsymbol{x}_s)$ in $\boldsymbol{A}(t)$) and the generation of $\boldsymbol{B}(t)$. Consequently, the first step in any transient noise analysis method is to perform a transient noiseless simulation prior to the transient noise simulation itself. This implies solving (3.20) for $\boldsymbol{x}_s$ by transient analysis while storing $\boldsymbol{J}(\boldsymbol{x}_s)$ at every time point and then generating $\boldsymbol{B}(t)$ for all time points using $\boldsymbol{x}_s$. The linearization

of the system around $\boldsymbol{x}_s$ can be interpreted as a linearization around the trajectory in time that the solution to the system takes when there is no noise. This linearization approach differs from the linearization that occurs in frequency domain methods, where the circuit is linearized around an operating point or periodic state.

3.5.2 Generating White Noise in the Time Domain

Time domain white noise generation plays an important role in transient noise analysis methods which inject random noise sources in the circuit at every time point, namely Monte Carlo methods. In the frequency domain, white noise has a constant PSD, meaning that the PSD is frequency independent. In order to generate white noise in the time domain, the white noise signal generated in the time domain must be constructed in such a way as to approximate the constant PSD of white noise in the frequency domain. This can be achieved by modeling the white noise signal as a random pulse waveform [24].

Given the PSD of a noise source, its associated white noise signal can be generated in the time domain in the following way. In the time domain, a white noise signal $n(t)$ with bandwidth F_{max} can be approximated as the random pulse waveform

$$n(t) = A\,\eta(t, \Delta t), \tag{3.25}$$

where $\eta(t, \Delta t)$ is a random number with Gaussian probability distribution updated with time interval Δt and A is the noise signal amplitude given by

$$A = \sqrt{2SF_{max}}, \tag{3.26}$$

where S is the double-sided PSD of the noise source detailed in Section 3.2. Using the following two properties:

$$\lim_{T\to\infty} \frac{1}{T} \int_T n(\tau)n(\tau)d\tau = A^2 \tag{3.27}$$

and

$$\lim_{T\to\infty}\frac{1}{T}\int_{T} n(\tau)n(\Delta t-\tau)d\tau = 0, \tag{3.28}$$

the autocorrelation function of the white noise signal is then given by

$$n^2(t) = \lim_{T\to\infty}\frac{1}{T}\int_{T} n(\tau)n(t-\tau)d\tau = A^2\Lambda\left(\frac{t}{\Delta t}\right), \tag{3.29}$$

where $\Lambda\left(\frac{t}{\Delta t}\right)$ is a triangular pulse of width Δt. Next, the PSD of $n(t)$ can be obtained by calculating the Fourier transform of the autocorrelation function as follows:

$$n^2(f) = \int_{-\infty}^{\infty} n^2(t)e^{-j2\pi fx}dx = A^2\Delta t\,\mathrm{sinc}^2(f\Delta t), \tag{3.30}$$

where $\mathrm{sinc}^2(y)$ is the normalized sinc function. The PSD of the generated white noise signal is shown in Fig. 3–3. As discussed in Section 3.3, F_{max}

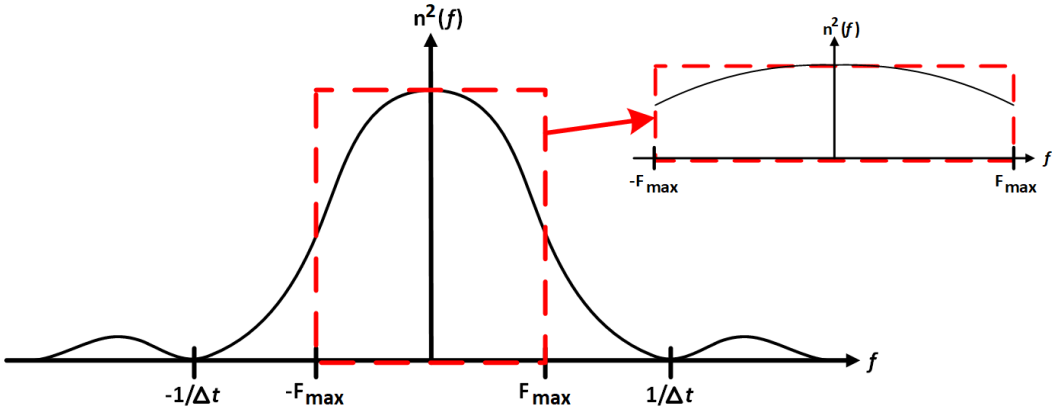

Figure 3–3: Double-sided PSD of white noise generated in the time domain (solid line) and the approximation of a constant PSD (dashed line) using the noise bandwidth F_{max}.

is carefully selected to fully capture high frequency noise from all the noise sources. Now, since the PSD in (3.30) is a function of Δt but not a function of F_{max}, the former must be selected in relation with the latter to ensure that the generated white noise signal is indeed a good approximation of a constant PSD. As shown in Fig. 3–3, for the PSD in (3.30) to be a good approximation

of the constant PSD of white noise, the upper bound for the time interval is

$$\Delta t = \frac{1}{2F_{max}}. \tag{3.31}$$

At this value of Δt or smaller, the generated white signal is a good approximation of the constant PSD of white noise up until the absolute value of the frequency approaches F_{max}. However, as discussed in Section 3.3, F_{max} is typically chosen such that its value is slightly higher than the largest individual bandwidth(s) out of all the noise sources in the circuit. This implies that the bulk of the noise lies at frequencies much lower than F_{max}, which ensures that the generated white signal is a good approximation of the constant PSD of white noise.

Furthermore, as mentioned in Section 3.3, there is a trade-off between noise simulation accuracy and overall noise simulation time through careful selection of F_{max} for white noise generation in the time domain. Assuming a fixed value of Δt in relation to F_{max} in (3.31), F_{max} may be decreased to increase Δt, thereby lowering the overall noise simulation time. However, doing so affects how good of an approximation of a constant PSD (3.30) is and can even make the noise simulation completely fail to capture high-frequency white noise in the circuit if the F_{max} falls below the largest individual bandwidth(s) out of all the noise sources in the circuit.

3.5.3 Monte Carlo Method

Time domain noise analysis is traditionally based on the Monte Carlo method, in which a large number of transient noise simulations are performed to represent the many different paths taken by the randomly generated noise sources [3]. As the number of Monte Carlo simulations increases, the solution to the noise analysis problem approaches its expected value by virtue of the law of large numbers in probability theory. More precisely, the mean and the variance of the noise in the circuit as a function of time may be

estimated through Monte Carlo transient noise analysis if enough Monte Carlo simulations are performed. The Monte Carlo method can be applied to the MNA formulation of the circuit equations which include the noise sources as follows. Let the superscript (i) denote the i^{th} Monte Carlo iteration, where $i = 0, 1, \ldots, m$. The formulation in (3.24) then becomes:

$$\boldsymbol{A}(t)\,\boldsymbol{x}_{noise}^{(i)} + \boldsymbol{C}\,\dot{\boldsymbol{x}}_{noise}^{(i)} + \boldsymbol{B}(t)\,\boldsymbol{v}^{(i)} \cong 0 \qquad \boldsymbol{x}_{noise}^{(i)}(0) = \boldsymbol{x}_{noise,0}^{(i)}, \tag{3.32}$$

where the product of $\boldsymbol{B}(t)\,\boldsymbol{v}^{(i)}$ contains the noise signals defined in (3.25). More specifically, the noise signal amplitudes are placed in $\boldsymbol{B}(t)$ as detailed in subsection 3.5.1 and the corresponding random numbers with Gaussian probability distribution are placed in $\boldsymbol{v}^{(i)}$.

As discussed in Section 3.5.1, the first step in performing a transient noise simulation of a given circuit using the Monte Carlo method is to perform a single transient noiseless simulation of the circuit in order to find $\boldsymbol{B}(t)$ and, in the case where the circuit is nonlinear, $\boldsymbol{A}(t)$. This transient noiseless simulation is a nonlinear transient simulation if the circuit is nonlinear or a linear transient simulation if the circuit is linear. Regardless of whether or not the circuit is linear, the second step is to solve (3.32) m times using *linear* transient analysis. Since $\boldsymbol{v}^{(i)}$ is different for each Monte Carlo iteration, the solution $\boldsymbol{x}_{noise}^{(i)}$ will also be different for each Monte Carlo iteration.

Now, considering that $\boldsymbol{x}_{noise}$ is the noise in the solution, the mean and the variance of the noise in the circuit can be computed by taking the mean and the variance of $\boldsymbol{x}_{noise}^{(i)}$ as a function of time across the i^{th} dimension. It should be noted that for the noise simulation to be accurate, $\boldsymbol{v}^{(i)}$ must be generated using a random number generator in such a way as to have a mean of 0 and a variance of 1 across the i^{th} dimension. Moreover, it may be helpful to randomly generate all the values of $\boldsymbol{v}^{(i)}$ in advance and then normalize

their mean and variance across the i^{th} dimension to ensure that they indeed have a mean of 0 and a variance of 1. However, due to the added memory cost, this should only be done for simulations with a relatively low number of Monte Carlo simulations. As the number of Monte Carlo simulations becomes relatively high, the mean and variance of the randomly generated numbers will approach their true value, thereby eliminating the need to normalize the mean and variance of $\boldsymbol{v}^{(i)}$.

Furthermore, since the Monte Carlo method relies on the generation of white noise at every time point to model thermal and shot noise, accurate Monte Carlo simulations require the use of a very small time step as discussed in subsection 3.5.2. This results in highly inefficient transient noise simulations that need to be repeated in a Monte Carlo fashion, which leads to a very high overall CPU cost. That being said, because Monte Carlo simulations are independent of each other, computing them in parallel to greatly reduce computation time is straightforward. Each Monte Carlo iteration of (3.32) can be solved by a given processor and the only time where the processors need to communicate is during the calculation of the statistics of the noise at the very end of the simulation. Nevertheless, the CPU cost of Monte Carlo noise simulation, even after parallelization, is still high.

Finally, accurate transient noise analysis requires long individual transient noise simulations that are performed a very large number of times in a Monte Carlo fashion, resulting in significant memory requirements. Memory issues mainly occur during the calculation of the variance, as every single Monte Carlo simulation is stored in memory until then due to the way variance is calculated. However, there exist some methods which can progressively and accurately compute the variance as new samples become available without the need to store all the samples in memory [5]. Unfortunately, due to their

sequential nature, these methods are ill-suited for Monte Carlo based transient noise analysis methods that are implemented in parallel.

3.5.4 Non-Monte Carlo Method

A non-Monte Carlo method is a transient noise analysis method that avoids the main drawbacks of the Monte Carlo method, more precisely the need for a very small time step combined with a large number of transient noise simulations performed in a Monte Carlo fashion. It does so by making use of the theory of SDEs to directly compute the mean and correlation matrix and hence the variance of the noise without having to outright inject noise sources at every time point. However, the noise system needs to be remodeled before the theory of SDEs can be employed for such a purpose. The following derivations describe how this can be done and are detailed in [7].

Derivation of the System of Linear SDEs

The objective of this initial derivation is to remodel the noise system in (3.24) as a system of linear SDEs of the form:

$$\dot{\boldsymbol{y}} = \boldsymbol{E}(t)\,\boldsymbol{y} + \boldsymbol{F}(t)\,\boldsymbol{u} \qquad \boldsymbol{y}(0) = \boldsymbol{y}_0. \tag{3.33}$$

At first glance, this can be achieved by multiplying both sides of (3.24) by the inverse of the $\boldsymbol{C}$ matrix. However, this assumes that $\boldsymbol{C}$ is a full-rank matrix and hence is invertible, which is generally not the case as it often has zero rows and columns. For instance, all of the entries in the first two rows and columns of $\boldsymbol{C}$ in (2.4) are zero because the two nodes which are represented by the node voltages $v_1(t)$ and $v_2(t)$ do not have any capacitors connected to them in the circuit shown in Fig. 2–1. In MNA formulations for which the contribution from the circuit elements to $\boldsymbol{C}$ is always symmetric, the number of zero rows is equal to the number of zero columns.

To ensure that $\boldsymbol{C}$ is a full-rank matrix, the circuit variables in $\boldsymbol{x}_{noise}$ are reordered in such a way as to regroup the zero columns of $\boldsymbol{C}$ on the right side of the matrix and the equations are reordered in such a way as to regroup the zero rows of $\boldsymbol{C}$ at the bottom of the matrix, which results in (3.24) taking the following form:

$$\begin{bmatrix} \boldsymbol{A}_{11}(t) & \boldsymbol{A}_{12}(t) \\ \boldsymbol{A}_{21}(t) & \boldsymbol{A}_{22}(t) \end{bmatrix} \begin{bmatrix} \boldsymbol{x}^1_{noise} \\ \boldsymbol{x}^2_{noise} \end{bmatrix} + \begin{bmatrix} \boldsymbol{C}_{11} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{x}}^1_{noise} \\ \dot{\boldsymbol{x}}^2_{noise} \end{bmatrix} + \begin{bmatrix} \boldsymbol{B}_1(t) \\ \boldsymbol{B}_2(t) \end{bmatrix} \boldsymbol{v} \cong 0$$
$$\begin{bmatrix} \boldsymbol{x}^1_{noise}(0) \\ \boldsymbol{x}^2_{noise}(0) \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}^1_{noise,0} \\ \boldsymbol{x}^2_{noise,0} \end{bmatrix}, \tag{3.34}$$

where $\boldsymbol{A}_{11}(t) \in \mathbb{R}^{m\times m}$, $\boldsymbol{C}_{11} \in \mathbb{R}^{m\times m}$, $\boldsymbol{A}_{22}(t) \in \mathbb{R}^{k\times k}$, $\boldsymbol{A}_{12}(t) \in \mathbb{R}^{m\times k}$, $\boldsymbol{A}_{21}(t) \in \mathbb{R}^{k\times m}$, $\boldsymbol{B}_1(t) \in \mathbb{R}^{m\times q}$, $\boldsymbol{B}_2(t) \in \mathbb{R}^{k\times q}$, $\boldsymbol{x}^1_{noise} \in \mathbb{R}^m$, $\boldsymbol{x}^2_{noise} \in \mathbb{R}^k$, m is the number of nonzero rows/columns in $\boldsymbol{C}$, and k is the number of zero rows/columns in $\boldsymbol{C}$. In MNA formulations for which the contribution from the circuit elements to $\boldsymbol{C}$ is not always symmetric, some adjustments to the formulation of the system may be required to ensure that the number of zero rows is equal to the number of zero columns. As a consequence of the partitioning of the matrices, (3.34) can be separated into two sets of m and k equations as follows:

$$\begin{aligned} \boldsymbol{A}_{11}(t)\,\boldsymbol{x}^1_{noise} + \boldsymbol{A}_{12}(t)\,\boldsymbol{x}^2_{noise} + \boldsymbol{C}_{11}\,\dot{\boldsymbol{x}}^1_{noise} + \boldsymbol{B}_1(t)\,\boldsymbol{v} &= 0 \\ \boldsymbol{x}^1_{noise}(0) = \boldsymbol{x}^1_{noise,0} \qquad \boldsymbol{x}^2_{noise}(0) &= \boldsymbol{x}^2_{noise,0} \end{aligned} \tag{3.35}$$

and

$$\begin{aligned} \boldsymbol{A}_{21}(t)\,\boldsymbol{x}^1_{noise} + \boldsymbol{A}_{22}(t)\,\boldsymbol{x}^2_{noise} + \boldsymbol{B}_2(t)\,\boldsymbol{v} &= 0 \\ \boldsymbol{x}^1_{noise}(0) = \boldsymbol{x}^1_{noise,0} \qquad \boldsymbol{x}^2_{noise}(0) &= \boldsymbol{x}^2_{noise,0}. \end{aligned} \tag{3.36}$$

By solving for it in (3.36), $\boldsymbol{x}^2_{noise}$ is given by

$$\boldsymbol{x}^2_{noise} = -\left(\boldsymbol{A}_{22}(t)\right)^{-1} \boldsymbol{A}_{21}(t)\,\boldsymbol{x}^1_{noise} - \left(\boldsymbol{A}_{22}(t)\right)^{-1} \boldsymbol{B}_2(t)\,\boldsymbol{v}, \tag{3.37}$$

which assumes that $\boldsymbol{A}_{22}(t)$ is invertible. This is usually true unless both nodes of a voltage source are connected to capacitors or one of them is connected to a capacitor while the other one is grounded, in which case the problem can be circumvented through manipulation of the circuit variables and equations in (3.24) [7].

Next, defining

$$\begin{aligned} \boldsymbol{D}_1(t) &= -\left(\boldsymbol{A}_{22}(t)\right)^{-1} \boldsymbol{A}_{21}(t) \\ \boldsymbol{D}_2(t) &= -\left(\boldsymbol{A}_{22}(t)\right)^{-1} \boldsymbol{B}_2(t) \end{aligned} \tag{3.38}$$

and then substituting (3.37) and (3.38) in (3.35) results in

$$\begin{aligned} \boldsymbol{A}_{11}(t)\, \boldsymbol{x}^1_{noise} + \boldsymbol{A}_{12}(t)\, \boldsymbol{D}_1(t)\, \boldsymbol{x}^1_{noise} + \boldsymbol{C}_{11}\, \dot{\boldsymbol{x}}^1_{noise} \\ + \boldsymbol{B}_1(t)\, \boldsymbol{v} + \boldsymbol{A}_{12}(t)\, \boldsymbol{D}_2(t)\, v = 0 \\ \boldsymbol{x}^1_{noise}(0) = \boldsymbol{x}^1_{noise,0}. \end{aligned} \tag{3.39}$$

Afterwards, defining

$$\begin{aligned} \tilde{\boldsymbol{E}}(t) &= -(\boldsymbol{A}_{11}(t) + \boldsymbol{A}_{12}(t)\, \boldsymbol{D}_1(t)) \\ \tilde{\boldsymbol{F}}(t) &= -(\boldsymbol{B}_1(t) + \boldsymbol{A}_{12}(t)\, \boldsymbol{D}_2(t)) \end{aligned} \tag{3.40}$$

and then substituting (3.40) in (3.39) while also rearranging the terms in (3.39) such that all the terms but the one which contains a time derivative are regrouped on the right-hand side of the equation leads to

$$\boldsymbol{C}_{11}\, \dot{\boldsymbol{x}}^1_{noise} = \tilde{\boldsymbol{E}}(t)\, \boldsymbol{x}^1_{noise} + \tilde{\boldsymbol{F}}(t)\, \boldsymbol{v} \qquad \boldsymbol{x}^1_{noise}(0) = \boldsymbol{x}^1_{noise,0}. \tag{3.41}$$

Finally, a system of linear SDEs in the form of (3.33) can be obtained by multiplying both sides of (3.41) by the inverse of $\boldsymbol{C}_{11}$ as follows:

$$\dot{\boldsymbol{x}}^1_{noise} = \boldsymbol{E}(t)\, \boldsymbol{x}^1_{noise} + \boldsymbol{F}(t)\, \boldsymbol{v} \qquad \boldsymbol{x}^1_{noise}(0) = \boldsymbol{x}^1_{noise,0}, \tag{3.42}$$

where $\boldsymbol{E}(t) \in \mathbb{R}^{m \times m}$ and $\boldsymbol{F}(t) \in \mathbb{R}^{m \times q}$ are respectively equal to $\tilde{\boldsymbol{E}}(t)$ and $\tilde{\boldsymbol{F}}(t)$ multiplied by the inverse of $\boldsymbol{C}_{11}$ from the left. This assumes that $\boldsymbol{C}_{11}$ is

invertible, which is true so long as every node in the circuit that is connected to a capacitor has a capacitive path to ground or an independent voltage source node. It is therefore often necessary to model the parasitic capacitances in a circuit to ensure that this condition is respected.

Derivation of the System of ODEs for the Noise Correlation Matrix

Now that the noise system in (3.42) is in a suitable form, the theory of SDEs can be used to formulate the equations which govern the mean and the correlation matrix of $\boldsymbol{x}^1_{noise}$. Consider the differential form of (3.42) given by

$$d\boldsymbol{x}^1_{noise} = \boldsymbol{E}(t)\,\boldsymbol{x}^1_{noise}\,dt + \boldsymbol{F}(t)\,d\boldsymbol{w} \qquad \boldsymbol{x}^1_{noise}(0) = \boldsymbol{x}^1_{noise,0}, \tag{3.43}$$

where $\boldsymbol{w}$ is a vector of q independent Wiener processes defined as

$$d\boldsymbol{w} = \boldsymbol{v}\,dt. \tag{3.44}$$

For a system of linear SDEs with additive noise of the form of (3.43), the exact solution is given by

$$\boldsymbol{x}^1_{noise}(t) = \boldsymbol{\Phi}(t,t_0)\,\boldsymbol{x}^1_{noise}(t_0) + \int_{t_0}^{t} \boldsymbol{\Phi}(t,\tau)\,\boldsymbol{F}(\tau)\,d\boldsymbol{w}(\tau), \tag{3.45}$$

where $\boldsymbol{\Phi}(t,t_0)$ is the state-transition matrix of the system which corresponds to the solution of

$$\dot{\boldsymbol{\Phi}}(t,t_0) = \boldsymbol{E}(t)\,\boldsymbol{\Phi}(t,t_0) \qquad \boldsymbol{\Phi}(t_0,t_0) = \boldsymbol{I} \tag{3.46}$$

and I is the identity matrix [1]. Again, the main objective in time domain noise simulation is to find the mean and variance of the noise in the circuit as a function of time, which in this case implies finding the mean and the correlation matrix of $\boldsymbol{x}^1_{noise}$ over the time interval of interest. In order to first

find the mean of $\boldsymbol{x}^1_{noise}$, let $t_0 = 0$ such that (3.45) simplifies to

$$\boldsymbol{x}^1_{noise}(t) = \boldsymbol{\Phi}(t,0)\,\boldsymbol{x}^1_{noise,0} + \int_0^t \boldsymbol{\Phi}(t,\tau)\,\boldsymbol{F}(\tau)\,d\boldsymbol{w}(\tau). \tag{3.47}$$

The mean of $\boldsymbol{x}^1_{noise}$ can then be obtained by taking the expectation of both sides of (3.47) while using (3.44) and noting that $\mathbb{E}\left[\boldsymbol{v}(t)\right] = 0$ and $\mathbb{E}\left[\boldsymbol{x}^1_{noise,0}\right] = 0$ as follows:

$$\begin{aligned}
\boldsymbol{\mu}^1(t) = \mathbb{E}\left[\boldsymbol{x}^1_{noise}(t)\right] &= \mathbb{E}\left[\boldsymbol{\Phi}(t,0)\,\boldsymbol{x}^1_{noise,0}\right] + \mathbb{E}\left[\int_0^t \boldsymbol{\Phi}(t,\tau)\,\boldsymbol{F}(\tau) dw(\tau)\right] \\
&= \boldsymbol{\Phi}(t,0)\,\mathbb{E}\left[\boldsymbol{x}^1_{noise,0}\right] + \int_0^t \boldsymbol{\Phi}(t,\tau)\,\boldsymbol{F}(\tau)\,\mathbb{E}\left[\boldsymbol{v}(\tau)\right] d\tau \\
&= 0.
\end{aligned} \tag{3.48}$$

This implies that the mean of the noise variables in $\boldsymbol{x}^1_{noise}$ is always zero for all kinds of circuits and for all time t under the current system formulation.

Next, the correlation matrix of $\boldsymbol{x}^1_{noise}$, which is defined as

$$\boldsymbol{K}^1(t) = \mathbb{E}\left[\boldsymbol{x}^1_{noise}(t)\,\boldsymbol{x}^1_{noise}(t)^T\right], \tag{3.49}$$

satisfies the differential Lyapunov equation

$$\dot{\boldsymbol{K}}^1(t) = \boldsymbol{E}(t)\,\boldsymbol{K}^1(t) + \boldsymbol{K}^1(t)\,\boldsymbol{E}(t)^T + \boldsymbol{F}(t)\,\boldsymbol{F}(t)^T \qquad \boldsymbol{K}^1(0) = \boldsymbol{K}^1_0, \tag{3.50}$$

where $\boldsymbol{K}^1(t) \in \mathbb{R}^{m\times m}$ is a symmetric positive semi-definite matrix [1]. The initial value $\boldsymbol{K}^1_0$ corresponds to the solution to the continuous Lyapunov equation

$$\boldsymbol{E}(0)\,\boldsymbol{K}^1_0 + \boldsymbol{K}^1_0\,\boldsymbol{E}(0)^T + \boldsymbol{F}(0)\,\boldsymbol{F}(0)^T = 0, \tag{3.51}$$

where $\boldsymbol{K}^1_0$ is assumed to be a symmetric positive semi-definite matrix [1]. This is generally true unless $\boldsymbol{E}(0)$ has eigenvalues with nonnegative real parts, in which case $\boldsymbol{K}^1_0$ is set to 0 [7]. Since $\boldsymbol{K}^1(t)$ represents the correlation matrix of $\boldsymbol{x}^1_{noise}$ as a function of time, the noise variance of the circuit variables in $\boldsymbol{x}^1_{noise}$

as a function of time can be obtained directly from the diagonal entries of $\boldsymbol{K}^1(t)$. Similarly, the noise correlations between the different circuit variables in $\boldsymbol{x}^1_{noise}$ as a function of time can be obtained directly from the non-diagonal entries of $\boldsymbol{K}^1(t)$. Considering $\boldsymbol{K}^1(t)$ is symmetric, (3.49) represents a system of $m(m+1)/2$ linear ODEs whose number quickly grows as the size of the circuit increases, with m being roughly equal to the number of nodes connected to a capacitor. The noise analysis problem, which originally involved solving a nonlinear stochastic system, is therefore reduced to solving a LTV system.

Numerical Computation of the Noise Correlation Matrix

The first step in finding the correlation matrix of $\boldsymbol{x}^1_{noise}$ as a function of time is to compute $\boldsymbol{E}(t)$ and $\boldsymbol{F}(t)$ numerically. This implies computing $\boldsymbol{A}(t)$ and $\boldsymbol{B}(t)$ numerically using the transient noiseless analysis solution at every time point as detailed in subsection 3.5.1 and then going through the derivations of the system of linear SDEs in (3.42). The second step is to obtain the initial value $\boldsymbol{K}^1_0$ by solving the continuous Lyapunov equation in (3.51) numerically. While the third step comprises the bulk of the computations, it can be bypassed for the class of circuits with time-invariant large-signal waveforms.

For circuits with time-invariant large-signal waveforms, $\boldsymbol{E}(t)$ and $\boldsymbol{F}(t)$ are constant with time. Consequently, the solution to (3.50) is given by

$$\boldsymbol{K}^1(t) = \boldsymbol{K}^1_0 \tag{3.52}$$

and hence the transient noise simulation reduces to solving the continuous Lyapunov equation in (3.51). This can be compared with the frequency domain noise simulation of a LTI circuit discussed in subsection 3.4.1 as both methods are applied to circuits with time-invariant large-signal waveforms. In fact, solving (3.51) is equivalent to calculating the noise variance in (3.6)

for every circuit variable in $\boldsymbol{x}^1_{noise}$ over the entire frequency spectrum (i.e. $F_{max} = \infty$).

For nonlinear circuits with arbitrary excitations, $\boldsymbol{E}(t)$ and $\boldsymbol{F}(t)$ are time-varying and the transient noise simulation is thus required to solve for $\boldsymbol{K}^1(t)$ in (3.50) as a function of time. To do so, (3.50) is discretized in time and written at time $t = t_{i+1}$ as follows:

$$\dot{\boldsymbol{K}}^1_{i+1} = \boldsymbol{E}_{i+1}\,\boldsymbol{K}^1_{i+1} + \boldsymbol{K}^1_{i+1}\,\boldsymbol{E}^T_{i+1} + \boldsymbol{F}_{i+1}\,\boldsymbol{F}^T_{i+1}, \tag{3.53}$$

where $\boldsymbol{K}^1_{i+1}$, $\boldsymbol{E}_{i+1}$, and $\boldsymbol{F}_{i+1}$ are respectively defined as $\boldsymbol{K}^1(t_{i+1})$, $\boldsymbol{E}(t_{i+1})$, and $\boldsymbol{F}(t_{i+1})$ for simplicity and $\boldsymbol{K}^1_0$ is the initial value that was obtained in the previous step. Using the backward Euler method to approximate the derivative in (3.53) results in the following difference equation:

$$\frac{\boldsymbol{K}^1_{i+1} - \boldsymbol{K}^1_i}{h} = \boldsymbol{E}_{i+1}\,\boldsymbol{K}^1_{i+1} + \boldsymbol{K}^1_{i+1}\,\boldsymbol{E}^T_{i+1} + \boldsymbol{F}_{i+1}\,\boldsymbol{F}^T_{i+1}, \tag{3.54}$$

where h is the time step that separates two successive time points. To solve for $\boldsymbol{K}^1_{i+1}$, all the terms are first regrouped on the left-hand side of (3.54) as follows:

$$\boldsymbol{E}_{i+1}\,\boldsymbol{K}^1_{i+1} - \frac{\boldsymbol{K}^1_{i+1}}{h} + \boldsymbol{K}^1_{i+1}\,\boldsymbol{E}^T_{i+1} + \boldsymbol{F}_{i+1}\,\boldsymbol{F}^T_{i+1} + \frac{\boldsymbol{K}^1_i}{h} = 0. \tag{3.55}$$

Next, $\frac{\boldsymbol{K}^1_{i+1}}{h}$ is expanded like so:

$$\boldsymbol{E}_{i+1}\,\boldsymbol{K}^1_{i+1} - \frac{1}{2h}\boldsymbol{I}\,\boldsymbol{K}^1_{i+1} - \frac{1}{2h}\boldsymbol{I}\,\boldsymbol{K}^1_{i+1} + \boldsymbol{K}^1_{i+1}\,\boldsymbol{E}^T_{i+1} + \boldsymbol{F}_{i+1}\,\boldsymbol{F}^T_{i+1} + \frac{\boldsymbol{K}^1_i}{h} = 0, \tag{3.56}$$

where $\boldsymbol{I}$ is the identity matrix. This allows (3.56) to then be factorized as follows:

$$\boldsymbol{P}\,\boldsymbol{K}^1_{i+1} + \boldsymbol{K}^1_{i+1}\,\boldsymbol{P}^T + \boldsymbol{Q} = 0, \tag{3.57}$$

where

$$\begin{aligned} \boldsymbol{P} &= \boldsymbol{E}_{i+1} - \frac{1}{2h}\boldsymbol{I} \\ \boldsymbol{Q} &= \boldsymbol{F}_{i+1}\,\boldsymbol{F}_{i+1}^{T} + \frac{\boldsymbol{K}_{i}^{1}}{h} \end{aligned} \tag{3.58}$$

and Q is symmetric. As $\boldsymbol{K}_{i+1}^{1}$ is symmetric, (3.57) represents a system of $m(m+1)/2$ linear equations.

Finally, (3.57) can be solved iteratively starting from $\boldsymbol{K}_{0}^{1}$ to obtain the correlation matrix of $\boldsymbol{x}_{noise}^{1}$ at every time point. Since (3.57) is a Lyapunov equation, it can be solved numerically using methods such as the Bartels-Stewart algorithm [2]. As this step requires solving a Lyapunov equation at every time point, the associated CPU cost becomes increasingly high as the size of the system increases. It should also be noted that the transient noiseless simulation and the transient noise simulation can be performed concurrently. This is because, contrary to the Monte Carlo method, there is only a single transient noise simulation. Consequently, the transient noise simulation is able to compute the solution at a certain time point once the transient noiseless simulation has performed the linearization of the system at that time point. Although the transient noiseless simulation is forced to take the time steps required by the transient noise simulation, the converse is not necessarily true.

3.6 Conclusion

In this chapter, various noise analysis methods are presented. Frequency domain methods are best suited for LTI or LPTV circuits operating under small-signal conditions, while time domain methods are preferred for nonlinear circuits containing arbitrary large-signal waveforms. The latter is traditionally based on the Monte Carlo method, which is associated with a high CPU cost, even if implemented in parallel, as well as potential memory issues. As for non-Monte Carlo methods, although they are typically faster than Monte Carlo

based methods, they can still be computationally expensive when the circuit is large as they require solving a Lyapunov equation at every time point.

CHAPTER 4
Parallel Non-Monte Carlo Transient Noise Analysis

4.1 Introduction

As seen in the previous chapter, non-Monte Carlo transient noise analysis appears to be an attractive option when it comes to time domain noise simulation of nonlinear circuits with large-signal waveforms due to the drawbacks of the Monte Carlo method. However, non-Monte Carlo transient noise analysis methods also suffer from a high CPU cost when the circuit is large and cannot seemingly be implemented in parallel to reduce computation time as is the case for Monte Carlo transient noise analysis methods. This is because non-Monte Carlo methods feature a single transient noise simulation whereas Monte Carlo methods rely on a multitude of transient noise simulations performed in a Monte Carlo fashion which are independent of each other. Yet, since non-Monte Carlo methods are typically faster than Monte Carlo methods, a transient noise analysis method that is even faster than the time domain methods detailed in the previous chapter would result from the implementation of a non-Monte Carlo method in parallel.

With this idea in mind, the present chapter proposes a novel approach to the implementation of a non-Monte Carlo method in parallel [11]. The achievement of this purpose begins with noticing that (3.50) represents a system of linear ODEs in the form of an initial value problem and that solving it accounts for the bulk of the computations in the transient noise simulation. The proposed method then allows for a parallel implementation of this computationally expensive step by using an approach similar to the *paraexp* algorithm, which is a parallel integrator for linear initial value problems

[9]. The algorithm was adapted to solve the Lyapunov equations resulting from the discretization of (3.50) in parallel even though they constitute a single transient noise simulation. Consequently, the CPU cost of the proposed method is lower than that of the non-Monte Carlo method that it is predicated upon, especially for large circuits. Additionally, the resulting parallel speedup is required to scale well with the number of processors used in the noise simulation and the size of the circuit.

4.2 Parallel Implementation of the Non-Monte Carlo Transient Noise Analysis Method

The parallel implementation of the non-Monte Carlo transient noise analysis method requires a more comprehensive examination of the equations which govern the noise correlation matrix. More precisely, it is the structure of these equations which allows for the single transient noise simulation to be parallelized in time. Because the differential Lyapunov equation in (3.50) is a linear system, the solution to it is given by

$$\boldsymbol{K}(t) = \boldsymbol{\Phi}(t,t_0)\,\boldsymbol{K}_0\,\boldsymbol{\Phi}(t,t_0)^T + \int_{t_0}^{t} \boldsymbol{\Phi}(t,\tau)\,\boldsymbol{F}(\tau)\,\boldsymbol{F}(\tau)^T\,\boldsymbol{\Phi}(t,\tau)^T\,d\tau, \qquad (4.1)$$

where $\boldsymbol{\Phi}(t,t_0) \in \mathbb{R}^{k\times k}$ is the state-transition which corresponds to the solution of

$$\dot{\boldsymbol{\Phi}}(t,t_0) = \boldsymbol{E}(t)\,\boldsymbol{\Phi}(t,t_0) \qquad \boldsymbol{\Phi}(t_0,t_0) = \boldsymbol{I} \qquad (4.2)$$

and $\boldsymbol{I}$ is the identity matrix. The details of the derivation are shown in Appendix A. It should be noted that the superscript 1 which denotes that $\boldsymbol{K}^1(t)$ is the correlation matrix of $\boldsymbol{x}^1_{noise}$ in (3.50) is omitted in this chapter for simplicity. Furthermore, (4.1) can be separated into a homogeneous response $\boldsymbol{K}_1(t)$ and an inhomogeneous response $\boldsymbol{K}_2(t)$ that are given by

$$\boldsymbol{K}_1(t) = \boldsymbol{\Phi}(t,t_0)\,\boldsymbol{K}_0\,\boldsymbol{\Phi}(t,t_0)^T \qquad (4.3)$$

and

$$\boldsymbol{K}_2(t) = \int_{t_0}^{t} \boldsymbol{\Phi}(t,\tau)\, \boldsymbol{F}(\tau)\, \boldsymbol{F}(\tau)^T\, \boldsymbol{\Phi}(t,\tau)^T\, d\tau. \tag{4.4}$$

Since it does not depend on $\boldsymbol{F}(t)$, which contains all the noise sources that act as inputs in the noise simulation, $\boldsymbol{K}_1(t)$ can be thought of as the zero input response of (3.50). Similarly, $\boldsymbol{K}_2(t)$ can be seen as the zero state response of (3.50) as it does not depend on the initial condition $\boldsymbol{K}_0$.

To employ p processors in the parallel task of computing $\boldsymbol{K}(t)$ in (4.1), the time interval of interest $[t_0, T]$, where $t \in [t_0, T]$, is partitioned into subintervals $[T_{j-1}, T_j]$ with $j = 1, \ldots, p$ and $t_0 = T_0 < T_1 < \ldots < T_p = T$. These subintervals need not be the same length of time, however it makes sense for their lengths to be similar to each other for the purpose of splitting the parallel task evenly and thus reducing the overall computation time. Time partitioning schemes have also been proposed for this purpose [9]. $\boldsymbol{K}_2(t)$ and $\boldsymbol{K}_1(t)$ can then be written for each subinterval as

$$\boldsymbol{K}_{2,j}(t) = \int_{T_{j-1}}^{t} \boldsymbol{\Phi}(t,\tau)\, \boldsymbol{F}(\tau)\, \boldsymbol{F}(\tau)^T\, \boldsymbol{\Phi}(t,\tau)^T\, d\tau \tag{4.5}$$

where $\boldsymbol{K}_{2,j}(T_{j-1}) = 0$ and $t \in [T_{j-1}, T_j]$, and

$$\boldsymbol{K}_{1,j}(t) = \boldsymbol{\Phi}(t, T_{j-1})\, \boldsymbol{K}_{2,j-1}(T_{j-1})\, \boldsymbol{\Phi}(t, T_{j-1})^T \tag{4.6}$$

where $\boldsymbol{K}_{2,0}(T_0) = \boldsymbol{K}_0$ and $t \in [T_{j-1}, T]$. The p inhomogeneous subproblems associated with (4.5) are completely decoupled due to their zero initial condition, whereas the p homogeneous subproblems associated with (4.6) are also decoupled and can be solved once the initial condition $\boldsymbol{K}_{2,j-1}(T_{j-1})$ is available. Fig. 4–1 illustrates the time partitions as well as the decoupling of the subproblems for $p = 5$. For instance, in the example shown in Fig. 4–1, a given processor would be tasked to solve the inhomogeneous subproblem from $t = T_1$ to $t = T_2$ starting from the initial condition $\boldsymbol{K}_{2,2}(T_1) = 0$, then that same processor

would be tasked to solve the associated homogeneous subproblem from $t = T_2$ to $t = T_5$ using the initial condition $\boldsymbol{K}_{2,2}(T_2)$ that was obtained from solving the inhomogeneous subproblem beforehand. As they are decoupled, the subproblems with indices $j = 1, \ldots, p$ can be solved in parallel. The total solution to the original problem can then be obtained by summation of all the subproblems due to the system being linear. A proof of the decoupling of the subproblems as well as the recovery of the total solution by summation of all the subproblems is provided in Appendix B. Moreover, this summation which occurs at the very end of the transient noise simulation is the only instance where the processors need to communicate, which helps limit the communication overhead. Despite that, a notable delay may occur before the end of the communication step should some parallel tasks be completed significantly sooner than others.

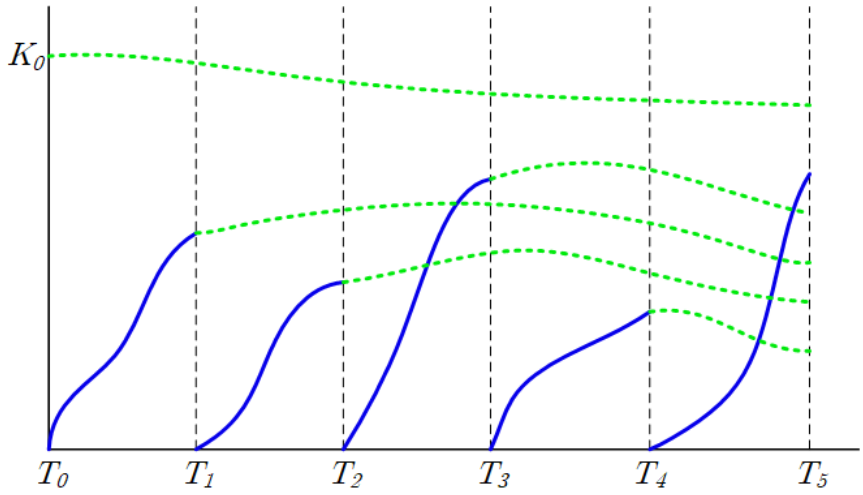

Figure 4–1: Overlapping time decomposition of (3.50) into five inhomogeneous subproblems with zero initial condition (solid curves) and five homogeneous subproblems (dotted curves) that are propagated from an initial condition.

4.3 Numerical Computation of the Solution to the Homogeneous and Inhomogeneous Subproblems

The process of computing the solution to the inhomogeneous subproblems varies little from the process of computing the solution to the original problem in (3.50). However, the process of computing the solution to the homogeneous subproblems differs greatly from the latter.

4.3.1 Solution to the Homogeneous Subproblems

Unlike solving for the solution to the original problem, solving for the solution to the homogeneous subproblems requires no explicit numerical integration. By making use of the state-transition matrix of the noisy system, initial conditions for the system described by the differential Lyapunov equation in (3.50) can be propagated quickly through time.

To first solve for the state-transition matrix, (4.2) is discretized in time and written at time $t = t_{i+1}$ as follows:

$$\dot{\boldsymbol{\Phi}}(t_{i+1}, t_0) = \boldsymbol{E}(t_{i+1})\, \boldsymbol{\Phi}(t_{i+1}, t_0) \qquad \boldsymbol{\Phi}(t_0, t_0) = \boldsymbol{I}. \tag{4.7}$$

Using the backward Euler method to approximate the derivative in (4.7) results in the following difference equation:

$$\frac{\boldsymbol{\Phi}(t_{i+1}, t_0) - \boldsymbol{\Phi}(t_i, t_0)}{h} = \boldsymbol{E}(t_{i+1})\, \boldsymbol{\Phi}(t_{i+1}, t_0) \qquad \boldsymbol{\Phi}(t_0, t_0) = \boldsymbol{I}, \tag{4.8}$$

where h is the time step which separates two successive time points. Solving for $\boldsymbol{\Phi}(t_{i+1}, t_0)$ in (4.8) leads to

$$\boldsymbol{\Phi}(t_{i+1}, t_0) = \left(\boldsymbol{I} - h\, \boldsymbol{E}(t_{i+1})\right)^{-1} \boldsymbol{\Phi}_i \qquad \boldsymbol{\Phi}(t_0, t_0) = \boldsymbol{I}. \tag{4.9}$$

Considering that the initial condition is the identity matrix, (4.9) can also be expressed for $i = 0, 1, 2, \ldots$ as

$$\boldsymbol{\Phi}(t_{i+1}, t_0) = \prod_{N=0}^{i} \left(\boldsymbol{I} - h\, \boldsymbol{E}(t_{N+1})\right)^{-1}. \tag{4.10}$$

The computation of the state-transition matrix thus only depends on the discretized values of $\boldsymbol{E}(t)$. However, since $\boldsymbol{E}(t)$ is generally time-varying, the transient noise simulation is required to compute the matrix inverse in (4.9) at every time point.

From (4.6), once the state-transition matrix $\boldsymbol{\Phi}(t_{i+1}, T_{j-1})$ at the current time point $t = t_{i+1}$ is available, the solution to the homogeneous subproblem $\boldsymbol{K}_{1,j}(t_{i+1})$ can be obtained by propagating the initial condition for the current subinterval $\boldsymbol{K}_{2,j-1}(T_{j-1})$ through time until the current time point by means of a multiplication with the state-transition matrix and its transpose. Intuitively, $\boldsymbol{K}_{2,j-1}(T_{j-1})$ is propagated progressively farther in time as time moves forward in the interval $[T_{j-1}, T]$ As this simply involves multiplying three matrices together, the main CPU cost involved in finding the homogeneous solution for a given subinterval arises from solving (4.9).

4.3.2 Solution to the Inhomogeneous Subproblems

As mentioned previously, solving for the inhomogeneous subproblems is akin to solving the original problem in (3.50). In fact, solving for the solution to the inhomogeneous subproblems corresponds to solving (3.57) for the current subinterval $[T_{j-1}, T_j]$ starting from zero initial condition. Using the notation in (4.5), (3.57) can be written at time $t = t_{i+1}$ for a given subinterval as

$$\boldsymbol{P}\,\boldsymbol{K}_{2,j}(t_{i+1}) + \boldsymbol{K}_{2,j}(t_{i+1})\,\boldsymbol{P}^T + \boldsymbol{Q} = 0, \tag{4.11}$$

where

$$\begin{aligned} \boldsymbol{P} &= \boldsymbol{E}(t_{i+1}) - \frac{1}{2h}\boldsymbol{I} \\ \boldsymbol{Q} &= \boldsymbol{F}(t_{i+1})\,\boldsymbol{F}(t_{i+1})^T + \frac{\boldsymbol{K}_{2,j}(t_i)}{h} \end{aligned} \tag{4.12}$$

The continuous Lyapunov equation in (4.11) is then solved iteratively through time starting from $\boldsymbol{K}_{2,j}(T_{j-1}) = 0$ for the current subinterval. As this involves once again solving a Lyapunov equation at every time point, the CPU cost

becomes increasingly high as the size of the system increases. However, since the proposed method computes all the inhomogeneous subproblems in parallel, the CPU cost for computing the inhomogeneous solution over the full time interval can be decreased significantly.

Moreover, the efficiency of the parallel algorithm relies on the homogeneous subproblems being much faster to solve than the inhomogeneous subproblems. One way to ensure that this is the case is to use a larger time step to solve the homogeneous subproblems than the time step used to solve the inhomogeneous subproblems and then interpolate the values of the homogeneous solutions at the time points which were initially disregarded. For instance, using a time step that is three times larger would require the values of the homogeneous solutions at two time points between two consecutive known values to be interpolated. While the number of interpolated values has an effect on the accuracy of the transient noise simulation, this effect is limited because of how the homogeneous solutions are propagated quickly over long time intervals as opposed to the inhomogeneous solutions which are solved through high accuracy integration on short subintervals.

4.4 Numerical Example

The proposed transient noise analysis method was tested on the bipolar junction transistor (BJT) low-noise amplifier (LNA) shown in Fig. 4–2 with a sinusoidal input at 300MHz. Only thermal and shot noise were considered in this example. Although the LNA circuit is fairly small, it is large enough in that the Lyapunov equations become sufficiently expensive to solve computationally for a speedup to be observed. The proposed method was compared to the base serial method in which the Lyapunov equations are solved using MATLAB's SLICOT routine SB03MD that is based upon the algorithm developed in [2]. Additionally, a comparison was made to the brute force Monte Carlo approach

implemented in parallel with the same number of processors. A contrast of the noise variance waveforms at the output of the amplifier is displayed in Fig. 4–3. The noise variance varies sinusoidally with time since the magnitude of the shot noise generated by the BJTs follows the sinusoidal input signal, while the average value of the noise variance can be attributed to the combined effect of thermal and shot noise. As observed in Fig. 4–3, the noise variance waveforms produced by the proposed method and the base serial method nearly overlap, while that of the Monte Carlo based method varies slightly from the other two. It is worth noting that if the amount of Monte Carlo simulations was taken to a number much larger than sixty thousand, then its resulting noise variance waveform would likely overlap much more closely with the other two waveforms due to the nature of the Monte Carlo method. However, the added memory and computational costs incurred by such an effort would be very high as well.

Furthermore, a summary of the computational costs using a server with 8 processors is shown in Table 4–1. The root mean square (RMS) error section in Table 4–1 supports the earlier claim that the difference between the Monte Carlo based method and the other two methods decreases as the number of Monte Carlo simulations increases and hence the variance waveform approaches its expected value due to the law of large numbers in probability theory. This section in Table 4–1 also features the RMS error between the proposed method and the base serial method, which can be compared to the scale of the waveforms in Fig. 4–3 to confirm that the proposed method is very close to the base serial method in terms of accuracy. Also shown in Table 4–1 is the vast difference in computation time between the Monte Carlo based method and the other two methods. This large gap in computation time would only be made even more significant should the number of Monte

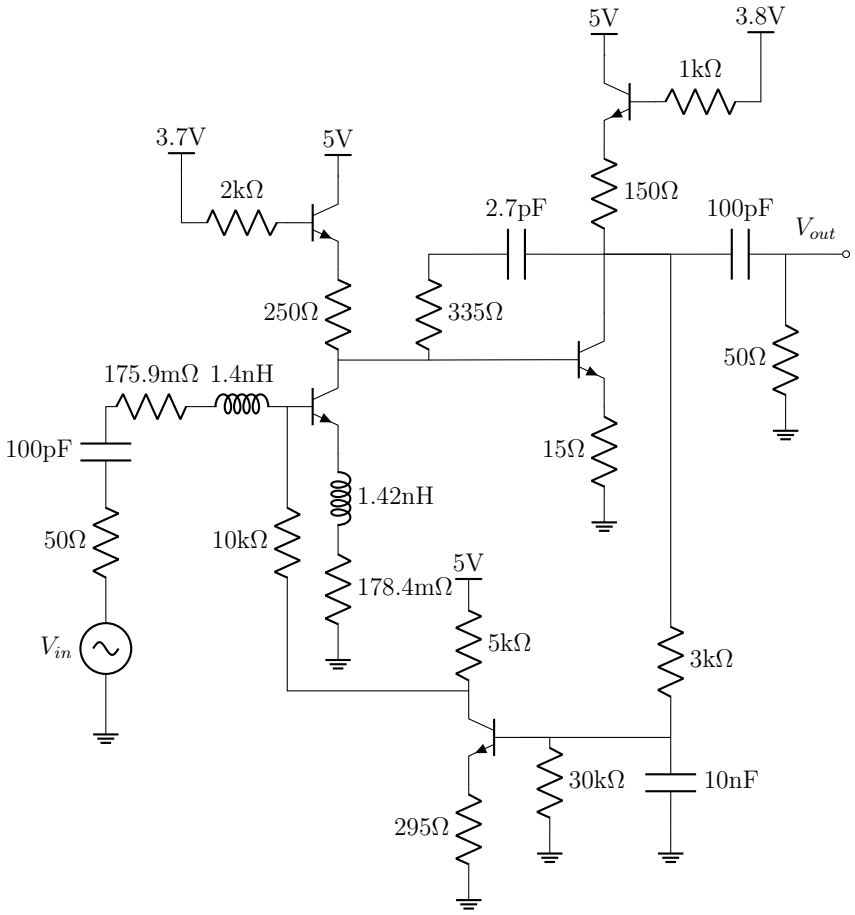

Figure 4–2: Low-Noise Amplifier Schematic [18].

Carlo simulations be increased further. As mentioned in subsection 3.5.3, the CPU cost of Monte Carlo based methods, even after parallelization, is still high. More importantly, the speedup achieved by the proposed method over the base serial method is indicated in Table 4–1. The speedup of 2.03 implies that the proposed parallel implementation of the base serial method speeds it up by a factor of 2.03 for the LNA circuit.

Lastly, Table 4–2 shows how the speedup increases as the number of processors increases. As observed in Table 4–2, the proposed method scales well with up to 8 processors used in the noise simulation. Scalability with the number of processors is a desirable feature for the proposed method from the perspective of large-scale applications. Yet, the proposed method cannot avoid

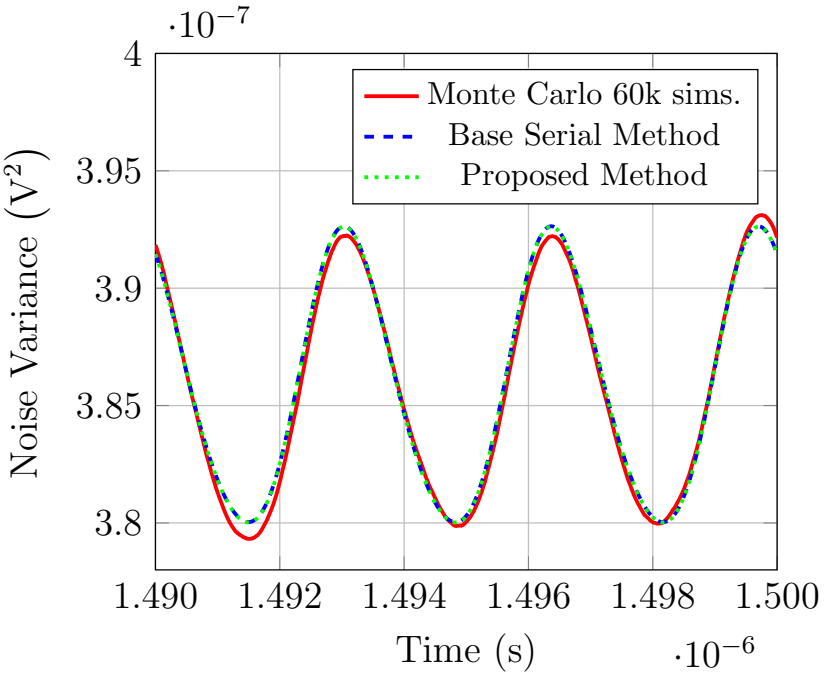

Figure 4–3: Comparison of the noise variance at the output of the LNA as a function of time due to thermal and shot noise.

the diminishing return in speedup that is characteristic of parallel algorithms. This diminishing return can be attributed partly to Amdahl's law in computer architecture, which states that the theoretical parallel speedup is limited by the serial part of the algorithm [28]. Additionally, the speedup is limited by the step involving the summation of all the solutions at the end of the noise simulation where the processors need to communicate. This is because the workload is not equally split between the processors as can be seen in Fig. 4–1 and the algorithm thus has to wait for all the parallel tasks to be completed before computing the total solution. The impact of this issue on speedup can be minimized by ensuring that the homogeneous solutions can be solved much quicker than the inhomogeneous solutions as discussed in Section 4.3.

4.5 Conclusion

In this chapter, the proposed parallel non-Monte Carlo transient noise analysis method is presented. The proposed method separates the differential

Table 4–1: Computation time, speedup, and output RMS error evaluated across the proposed method and other transient noise analysis methods for the LNA.

Method	**Monte Carlo 60k sims.**	**Base Serial Method**	**Proposed Method**
Computation Time (s)	27394.62	193.87	95.58
Speedup Compared to	**Monte Carlo 60k sims.**	$141.30\times$	$286.61\times$
	Base Serial Method	-	$2.03\times$
RMS Error Compared to	**Monte Carlo 20k sims.**	9.71×10^{-10}	9.64×10^{-10}
	Monte Carlo 40k sims.	5.87×10^{-10}	5.82×10^{-10}
	Monte Carlo 60k sims.	4.34×10^{-10}	4.25×10^{-10}
	Base Serial Method	-	1.62×10^{-11}

Table 4–2: Speedup achieved by the proposed method over the base serial method as a function of the number of processors.

Number of Processors	**1**	**2**	**4**	**6**	**8**
Speedup	$0.74\times$	$1.32\times$	$1.62\times$	$1.90\times$	$2.03\times$

Lyapunov equation into homogeneous and inhomogeneous subproblems that can be solved independently. Furthermore, the time interval is partitioned into subintervals where the homogeneous and inhomogeneous solutions can be computed in parallel to decrease computation time. When tested on a LNA circuit, the proposed method achieved an increasing parallel speedup over the base serial method as more processors were used in the simulation, culminating in a speedup of 2.03 using 8 processors.

CHAPTER 5
Conclusion

5.1 Summary

This thesis details various methods of addressing the noise analysis problem in the simulation and modeling of microelectronic systems. Unlike frequency noise analysis methods, time domain noise analysis methods are well suited for simulating noise in nonlinear circuits with arbitrary large-signal waveforms. Among them are the traditional Monte Carlo based transient noise analysis methods, whose main disadvantage is their large CPU cost. To avoid this drawback, non-Monte Carlo transient noise analysis methods which avoid injecting noise randomly at every time point and instead directly model the noise characteristics by relying on the theory of SDEs are described. However, this approach is still susceptible to long computation times if the circuit is large due to the need to solve a Lyapunov equation at every time point in the transient noise simulation.

This thesis also proposes a novel parallel non-Monte Carlo transient noise analysis method. By partitioning the time interval of interest into multiple independent subintervals, the proposed method is able to decouple the noise analysis problem into homogeneous and inhomogeneous subproblems which can be solved in parallel for each subinterval to decrease the overall noise simulation time.

5.2 Future Work

- Improving the efficiency at which the proposed method can solve the homogeneous subproblems. The overall efficiency of the proposed method relies on solving the homogeneous subproblems much quicker than solving

the inhomogeneous subproblems, which is not sufficiently the case with the current method. Consequently, the proposed method could achieve an even higher parallel speedup if this issue were to be addressed. One possible way to do so would be to introduce a selector vector that selects the proper output entries in the noise correlation matrix. That way, the state-transition matrix of the system could potentially be obtained through matrix-vector operations rather than matrix-matrix operations and the homogeneous solutions could thus be obtained quicker.

- Implement flicker noise in the proposed method. As indicated in the numerical example, only thermal and shot noise are considered at the moment. While flicker noise is considerably more difficult to implement than the other two types of noise, it ultimately does not change any part of the process through which noise analysis is performed in the proposed method once it has been properly modeled.

References

[1] S. S. Artemiev and T. A. Averina. *Numerical Analysis of Systems of Ordinary and Stochastic Differential Equations*. De Gruyter, 2011.

[2] R. H. Bartels and G. W. Stewart. Solution of the matrix equation ax + xb = c. *Commun. ACM*, 15(9):820–826, September 1972.

[3] P Bolcato and R Poujois. A new approach for noise simulation in transient analysis. In *[Proceedings] 1992 IEEE International Symposium on Circuits and Systems*, volume 2, pages 887–890. IEEE, 1992.

[4] John C. Butcher. *Numerical Methods for Ordinary Differential Equations*. John Wiley & Sons, New York, 2008.

[5] Tony F Chan, Gene H Golub, and Randall J LeVeque. Algorithms for computing the sample variance: Analysis and recommendations. *The American Statistician*, 37(3):242–247, 1983.

[6] Kwong-Shu Chao and Richard Saeks. Continuation methods in circuit analysis. *Proceedings of the IEEE*, 65(8):1187–1194, 1977.

[7] A. Demir, E. W. Y. Liu, and A. L. Sangiovanni-Vincentelli. Time-domain non-monte carlo noise simulation for nonlinear dynamic circuits with arbitrary excitations. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 15(5):493–505, 1996.

[8] Alper Demir and Alberto Sangiovanni-Vincentelli. *Analysis and simulation of noise in nonlinear electronic circuits and systems*, volume 425. Springer Science & Business Media, 2012.

[9] Martin J. Gander and Stefan Güttel. Paraexp: A parallel integrator for linear initial-value problems. *SIAM Journal on Scientific Computing*, 35(2):C123–C142, 2013.

[10] W. A. Gardner. *Introduction to Random Processes with Applications to Signals & Systems*. McGraw-Hill, New York, 2nd ed. edition, 1990.

[11] Alex Goulet, Mina Farhan, Marco T. Kassis, and Roni Khazaka. Parallel non-monte carlo transient noise simulation. *IEEE Microwave and Wireless Components Letters*, 31(6):634–637, 2021.

[12] Paul R Gray, Paul J Hurst, Stephen H Lewis, and Robert G Meyer. *Analysis and design of analog integrated circuits*. John Wiley & Sons, 2009.

[13] Chung-Wen Ho, Albert Ruehli, and Pierce Brennan. The modified nodal approach to network analysis. *IEEE Transactions on circuits and systems*, 22(6):504–509, 1975.

[14] FN Hooge and PA Bobbert. On the correlation function of $1/f$ noise. *Physica B: Condensed Matter*, 239(3-4):223–230, 1997.

[15] C.D. Hull and R.G. Meyer. A systematic approach to the analysis of noise in mixers. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 40(12):909–919, 1993.

[16] Christopher D Hull and Robert G Meyer. A systematic approach to the analysis of noise in mixers. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 40(12):909–919, 1993.

[17] Christopher Dennis Hull. *Analysis and optimization of monolithic RF downconversion receivers*. PhD thesis, University of California, Berkeley, 1992.

[18] Marco T. Kassis, Dani Tannir, Raffi Toukhtarian, and Roni Khazaka. Computation of the x-parameters of multi-tone circuits using multipoint moment expansion. In *2017 IEEE 26th Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS)*, 2017.

[19] K. Kellogg, L. Dunleavy, S. Skidmore, H. Morales, and C. White. Bridging the gap in noise spectral density measurements derived from flicker and noise figure measurement systems. In *2017 IEEE 18th Wireless and Microwave Technology Conference (WAMICON)*, pages 1–4, 2017.

[20] Marvin S Keshner. 1/f noise. *Proceedings of the IEEE*, 70(3):212–218, 1982.

[21] Roni Khazaka. *Projection based techniques for the simulation of RF circuits and high speed interconnects*. PhD thesis, Carleton University, 2002.

[22] T. H. Lee and A. Hajimiri. Oscillator phase noise: a tutorial. *IEEE Journal of Solid-State Circuits*, 35(3):326–336, 2000.

[23] A. G. Mahmutoglu and A. Demir. Non-monte carlo analysis of low-frequency noise: Exposition of intricate nonstationary behavior and comparison with legacy models. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(11):1825–1835, 2016.

[24] John A. McNeill. *Jitter in ring-oscillators*. PhD thesis, Boston University, Boston, MA, 1994.

[25] Mauro Parodi and Marco Storace. *Linear and nonlinear circuits: Basic & advanced concepts*, volume 1. Springer, 2018.

[26] B. Pellegrini, R. Saletti, B. Neri, and P. Terreni. $1/f^v$ noise generators. In *Noise in Physical Systems and* $1/f$ *Noise*. Elsevier, 1985.

[27] Stephen L Richter and Raymond A Decarlo. Continuation methods: Theory and applications. *IEEE Transactions on Systems, Man, and Cybernetics*, (4):459–464, 1983.

[28] David P Rodgers. Improvements in multiprocessor system design. *ACM SIGARCH Computer Architecture News*, 13(3):225–231, 1985.

[29] Youcef Saad and Martin H Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, 7(3):856–869, 1986.

[30] Jiri Vlach, Vlach Jiří, and Kishore Singhal. *Computer methods for circuit analysis and design*. Springer Science & Business Media, 1983.

APPENDIX A
Derivation of the Solution to the Differential Lyapunov Equation

The following derivation intends to prove that

$$\boldsymbol{K}(t) = \boldsymbol{\Phi}(t, t_0)\, \boldsymbol{K}_0\, \boldsymbol{\Phi}(t, t_0)^T + \int_{t_0}^{t} \boldsymbol{\Phi}(t, \tau)\, \boldsymbol{F}(\tau)\, \boldsymbol{F}(\tau)^T\, \boldsymbol{\Phi}(t, \tau)^T\, d\tau \qquad \text{(A.1)}$$

is the solution to the differential Lyapunov equation

$$\dot{\boldsymbol{K}}(t) = \boldsymbol{E}(t)\, \boldsymbol{K}(t) + \boldsymbol{K}(t)\, \boldsymbol{E}(t)^T + \boldsymbol{F}(t)\, \boldsymbol{F}(t)^T \qquad \boldsymbol{K}(0) = \boldsymbol{K}_0. \qquad \text{(A.2)}$$

It does so by first assuming the form of the solution in (A.1) to the differential Lyapunov equation in (A.2) and then differentiating it to produce the original differential Lyapunov equation.

Before proceeding with the differentiation, (A.1) is first segmented into two parts for simplicity denoted by

$$\boldsymbol{K}_1(t) = \boldsymbol{\Phi}(t, t_0)\, \boldsymbol{K}(0)\, \boldsymbol{\Phi}(t, t_0)^T \qquad \text{(A.3)}$$

and

$$\boldsymbol{K}_2(t) = \int_{t_0}^{t} \boldsymbol{\Phi}(t, \tau)\, \boldsymbol{F}(\tau)\, \boldsymbol{F}(\tau)^T\, \boldsymbol{\Phi}(t, \tau)^T\, d\tau. \qquad \text{(A.4)}$$

The derivative of (A.3) is then given by

$$\dot{\boldsymbol{K}}_1(t) = \dot{\boldsymbol{\Phi}}(t, t_0)\, \boldsymbol{K}(0)\, \boldsymbol{\Phi}(t, t_0)^T + \boldsymbol{\Phi}(t, t_0)\, \boldsymbol{K}(0)\, \dot{\boldsymbol{\Phi}}(t, t_0)^T. \qquad \text{(A.5)}$$

However, the state-transition matrix $\boldsymbol{\Phi}(t, t_0)$ is defined as the solution to the system

$$\dot{\boldsymbol{\Phi}}(t, t_0) = \boldsymbol{E}(t)\, \boldsymbol{\Phi}(t, t_0) \qquad \boldsymbol{\Phi}(t_0, t_0) = \boldsymbol{I}. \qquad \text{(A.6)}$$

Thus, (A.5) can be further simplified using (A.6) to

$$\dot{\boldsymbol{K}}_1(t) = \boldsymbol{E}(t)\,\boldsymbol{\Phi}(t,t_0)\,\boldsymbol{K}(0)\,\boldsymbol{\Phi}(t,t_0)^T + \boldsymbol{\Phi}(t,t_0)\,\boldsymbol{K}(0)\,\boldsymbol{\Phi}(t,t_0)^T\,\boldsymbol{E}(t)^T. \quad \text{(A.7)}$$

Next, using the Leibniz integral rule, the derivative of (A.4) is given by

$$\begin{aligned}\dot{\boldsymbol{K}}_2(t) = &\int_{t_0}^{t} \left[\dot{\boldsymbol{\Phi}}(t,\tau)\,\boldsymbol{F}(\tau)\,\boldsymbol{F}(\tau)^T\,\boldsymbol{\Phi}(t,\tau)^T + \boldsymbol{\Phi}(t,\tau)\,\boldsymbol{F}(\tau)\,\boldsymbol{F}(\tau)^T\,\dot{\boldsymbol{\Phi}}(t,\tau)^T\right] d\tau \\ &+ \boldsymbol{\Phi}(t,t)\,\boldsymbol{F}(t)\,\boldsymbol{F}(t)^T\,\boldsymbol{\Phi}(t,t)^T\,\frac{dt}{dt} \\ &- \boldsymbol{\Phi}(t,t_0)\,\boldsymbol{F}(t_0)\,\boldsymbol{F}(t_0)^T\,\boldsymbol{\Phi}(t,t_0)^T\,\frac{dt_0}{dt},\end{aligned} \quad \text{(A.8)}$$

which can be further simplified by using (A.6) to

$$\begin{aligned}\dot{\boldsymbol{K}}_2(t) = &\int_{t_0}^{t} \boldsymbol{E}(t)\,\boldsymbol{\Phi}(t,\tau)\,\boldsymbol{F}(\tau)\,\boldsymbol{F}(\tau)^T\,\boldsymbol{\Phi}(t,\tau)^T\,d\tau \\ &+ \int_{t_0}^{t} \boldsymbol{\Phi}(t,\tau)\,\boldsymbol{F}(\tau)\,\boldsymbol{F}(\tau)^T\,\boldsymbol{\Phi}(t,\tau)^T\,\boldsymbol{E}(t)^T\,d\tau + \boldsymbol{F}(t)\,\boldsymbol{F}(t)^T,\end{aligned} \quad \text{(A.9)}$$

where the integral has been split into the sum of its two parts in anticipation of the final step of the derivation.

Lastly, by recombining (A.7) and (A.9) and taking $\boldsymbol{E}(t)$ out of the integrals as it does not depend on τ, the derivative of (A.1) is given by

$$\begin{aligned}\dot{\boldsymbol{K}}(t) = &\boldsymbol{E}(t)\,\boldsymbol{\Phi}(t,t_0)\,\boldsymbol{K}(0)\,\boldsymbol{\Phi}(t,t_0)^T + \boldsymbol{E}(t)\int_{t_0}^{t} \boldsymbol{\Phi}(t,\tau)\,\boldsymbol{F}(\tau)\,\boldsymbol{F}(\tau)^T\,\boldsymbol{\Phi}(t,\tau)^T d\tau \\ &+ \boldsymbol{\Phi}(t,t_0)\,\boldsymbol{K}(0)\,\boldsymbol{\Phi}(t,t_0)^T\boldsymbol{E}(t)^T + \int_{t_0}^{t} \boldsymbol{\Phi}(t,\tau)\,\boldsymbol{F}(\tau)\,\boldsymbol{F}(\tau)^T\,\boldsymbol{\Phi}(t,\tau)^T d\tau\,\boldsymbol{E}(t)^T \\ &+ \boldsymbol{F}(t)\,\boldsymbol{F}(t)^T = \boldsymbol{E}(t)\,\boldsymbol{K}(t) + \boldsymbol{K}(t)\,\boldsymbol{E}(t)^T + \boldsymbol{F}(t)\,\boldsymbol{F}(t)^T,\end{aligned} \quad \text{(A.10)}$$

which is exactly the system in (A.2) and thus completes the proof.

APPENDIX B
Proof of the Decomposition of the Differential Lyapunov Equation into Homogeneous and Inhomogeneous Subproblems

The objective of this proof is twofold: it first demonstrates the ability to decompose the solution to the differential Lyapunov equation in (3.50) into homogeneous and inhomogeneous solutions where the inhomogeneous solutions provide the initial conditions for the associated homogeneous solutions, and then it shows that the total solution can be obtained by superposition of all the solutions.

The proof begins with recalling that the solution to the differential Lyapunov equation

$$\dot{\boldsymbol{K}}(t) = \boldsymbol{E}(t)\,\boldsymbol{K}(t) + \boldsymbol{K}(t)\,\boldsymbol{E}(t)^T + \boldsymbol{F}(t)\,\boldsymbol{F}(t)^T \qquad \boldsymbol{K}(t_0) = \boldsymbol{K}_0 \tag{B.1}$$

is given by

$$\boldsymbol{K}(t) = \boldsymbol{\Phi}(t,t_0)\,\boldsymbol{K}_0\,\boldsymbol{\Phi}(t,t_0)^T + \int_{t_0}^{t} \boldsymbol{\Phi}(t,\tau)\,\boldsymbol{F}(\tau)\,\boldsymbol{F}(\tau)^T\,\boldsymbol{\Phi}(t,\tau)^T\,d\tau \tag{B.2}$$

and that the solution in (B.2) can be further divided into a homogeneous response $\boldsymbol{K}_1(t)$ and an inhomogeneous response $\boldsymbol{K}_2(t)$ that correspond to

$$\boldsymbol{K}_1(t) = \boldsymbol{\Phi}(t,t_0)\,\boldsymbol{K}_0\,\boldsymbol{\Phi}(t,t_0)^T \tag{B.3}$$

and

$$\boldsymbol{K}_2(t) = \int_{t_0}^{t} \boldsymbol{\Phi}(t,\tau)\,\boldsymbol{F}(\tau)\,\boldsymbol{F}(\tau)^T\,\boldsymbol{\Phi}(t,\tau)^T\,d\tau. \tag{B.4}$$

Then, let $t \in [0,T]$ and partition the time interval $[0,T]$ into subintervals $[T_{j-1},T_j]$ with $j = 1,\ldots,p$ and $0 = t_0 = T_0 < T_1 < \ldots < T_p = T$. Writing

(B.2) at the last time point $t = T$ and separating the integral (i.e. the inhomogeneous response) over the full time interval into a sum of integrals over the subintervals results in the following:

$$\begin{aligned}
\boldsymbol{K}(T) &= \boldsymbol{\Phi}(T,0)\,\boldsymbol{K}_0\,\boldsymbol{\Phi}(T,0)^T + \int_0^T \boldsymbol{\Phi}(T,\tau)\,\boldsymbol{F}(\tau)\,\boldsymbol{F}(\tau)^T\,\boldsymbol{\Phi}(T,\tau)^T\,d\tau \\
&= \boldsymbol{\Phi}(T,0)\,\boldsymbol{K}_0\,\boldsymbol{\Phi}(T,0)^T + \int_0^{T_1} \boldsymbol{\Phi}(T,\tau)\,\boldsymbol{F}(\tau)\,\boldsymbol{F}(\tau)^T\,\boldsymbol{\Phi}(T,\tau)^T\,d\tau \\
&\quad + \int_{T_1}^{T_2} \boldsymbol{\Phi}(T,\tau)\,\boldsymbol{F}(\tau)\,\boldsymbol{F}(\tau)^T\,\boldsymbol{\Phi}(T,\tau)^T\,d\tau + \ldots \\
&\quad + \int_{T_{p-1}}^{T} \boldsymbol{\Phi}(T,\tau)\,\boldsymbol{F}(\tau)\,\boldsymbol{F}(\tau)^T\,\boldsymbol{\Phi}(T,\tau)^T\,d\tau.
\end{aligned} \tag{B.5}$$

Next, by using the properties of the state-transition matrix

$$\begin{aligned}
\boldsymbol{\Phi}(t_q,t_s) &= \boldsymbol{\Phi}(t_q,t_r)\,\boldsymbol{\Phi}(t_r,t_s) \\
\boldsymbol{\Phi}(t_q,t_s)^T &= \boldsymbol{\Phi}(t_r,t_s)^T\,\boldsymbol{\Phi}(t_q,t_r)^T,
\end{aligned} \tag{B.6}$$

where $q, r, s = 0, 1, 2, \ldots$, (B.5) can be expanded as follows:

$$\begin{aligned}
\boldsymbol{K}(T) &= \boldsymbol{\Phi}(T,0)\,\boldsymbol{K}_0\,\boldsymbol{\Phi}(T,0)^T \\
&\quad + \int_0^{T_1} \boldsymbol{\Phi}(T,T_1)\,\boldsymbol{\Phi}(T_1,\tau)\,\boldsymbol{F}(\tau)\,\boldsymbol{F}(\tau)^T\,\boldsymbol{\Phi}(T_1,\tau)^T\,\boldsymbol{\Phi}(T,T_1)^T\,d\tau \\
&\quad + \int_{T_1}^{T_2} \boldsymbol{\Phi}(T,T_2)\,\boldsymbol{\Phi}(T_2,\tau)\,\boldsymbol{F}(\tau)\,\boldsymbol{F}(\tau)^T\,\boldsymbol{\Phi}(T_2,\tau)^T\,\boldsymbol{\Phi}(T,T_2)^T\,d\tau + \ldots \\
&\quad + \int_{T_{p-1}}^{T} \boldsymbol{\Phi}(T,T)\,\boldsymbol{\Phi}(T,\tau)\,\boldsymbol{F}(\tau)\,\boldsymbol{F}(\tau)^T\,\boldsymbol{\Phi}(T,\tau)^T\,\boldsymbol{\Phi}(T,T)^T\,d\tau.
\end{aligned} \tag{B.7}$$

The state-transition matrices which do not have τ as an argument are then moved outside their respective integrals, which leads to

$$\begin{aligned}\boldsymbol{K}(T) = &\boldsymbol{\Phi}(T,0)\,\boldsymbol{K}_0\,\boldsymbol{\Phi}(T,0)^T \\ &+ \boldsymbol{\Phi}(T,T_1)\left(\int_0^{T_1}\boldsymbol{\Phi}(T_1,\tau)\,\boldsymbol{F}(\tau)\,\boldsymbol{F}(\tau)^T\,\boldsymbol{\Phi}(T_1,\tau)^T\,d\tau\right)\boldsymbol{\Phi}(T,T_1)^T \\ &+ \boldsymbol{\Phi}(T,T_2)\left(\int_{T_1}^{T_2}\boldsymbol{\Phi}(T_2,\tau)\,\boldsymbol{F}(\tau)\,\boldsymbol{F}(\tau)^T\,\boldsymbol{\Phi}(T_2,\tau)^T\,d\tau\right)\boldsymbol{\Phi}(T,T_2)^T + \ldots \\ &+ \boldsymbol{\Phi}(T,T)\left(\int_{T_{p-1}}^{T}\boldsymbol{\Phi}(T,\tau)\,\boldsymbol{F}(\tau)\,\boldsymbol{F}(\tau)^T\,\boldsymbol{\Phi}(T,\tau)^T\,d\tau\right)\boldsymbol{\Phi}(T,T)^T.\end{aligned} \tag{B.8}$$

Finally, noting that

$$\boldsymbol{\Phi}(t_q,t_q) = I \tag{B.9}$$

by definition, where I is the identity matrix, (B.8) can be simplified as follows:

$$\begin{aligned}\boldsymbol{K}(T) = &\boldsymbol{\Phi}(T,0)\,\boldsymbol{K}_0\,\boldsymbol{\Phi}(T,0)^T + \boldsymbol{\Phi}(T,T_1)\,\tilde{\boldsymbol{K}}_1(T_1)\,\boldsymbol{\Phi}(T,T_1)^T \\ &+ \boldsymbol{\Phi}(T,T_2)\,\tilde{\boldsymbol{K}}_2(T_2)\,\boldsymbol{\Phi}(T,T_2)^T + \ldots \\ &+ \boldsymbol{\Phi}(T,T_{p-1})\,\tilde{\boldsymbol{K}}_{p-1}(T_{p-1})\,\boldsymbol{\Phi}(T,T_{p-1})^T + \tilde{\boldsymbol{K}}_p(T),\end{aligned} \tag{B.10}$$

where

$$\tilde{\boldsymbol{K}}_i(u) = \begin{cases}\int_{T_{i-1}}^{u}\boldsymbol{\Phi}(t,\tau)\,\boldsymbol{F}(\tau)\,\boldsymbol{F}(\tau)^T\,\boldsymbol{\Phi}(t,\tau)^T\,d\tau & T_{i-1}\le u\le T_i \\ 0 & \text{otherwise}\end{cases} \tag{B.11}$$

is the solution to the inhomogeneous subproblem which belongs to the subinterval $[T_{i-1},T_i]$ and $i=1,2,\ldots,p$. The $\tilde{\boldsymbol{K}}_i$ are then propagated through time until the last time point of the time interval $t=T$ by multiplying them respectively by $\boldsymbol{\Phi}(T,T_i)$ and $\boldsymbol{\Phi}(T,T_i)^T$ from the left and from the right. This corresponds exactly to solving the associated homogeneous subproblems using the solution to the inhomogeneous subproblem at the last time point $\tilde{\boldsymbol{K}}_i(T_i)$ as an initial condition as discussed in Section 4.2. Moreover, the first and last term on

the right-hand side of (B.10) represent the purely homogeneous and purely inhomogeneous solutions as shown in Fig. 4–1.

The next step is to prove that the total solution $\boldsymbol{K}(t)$ at any given time t can be obtained by superposition of all the solutions at that point in time. According to Fig. 4–1, all the solutions at any point in time consist of one inhomogeneous solution and one or more homogeneous solutions, one of which is a purely homogeneous solution with initial condition $\boldsymbol{K}_0$. With this in mind, let $t \in [T_{k-1}, T_k]$, where $1 < k < p$, $k \in \mathbb{N}$, and $[T_{k-1}, T_k]$ is an arbitrary subinterval. The total solution in (B.10) at time t is then given by

$$\begin{aligned}\boldsymbol{K}(t) = &\boldsymbol{\Phi}(t,0)\,\boldsymbol{K}_0\,\boldsymbol{\Phi}(t,0)^T + \boldsymbol{\Phi}(t,T_1)\,\tilde{\boldsymbol{K}}_1(T_1)\,\boldsymbol{\Phi}(t,T_1)^T + \dots \\ &+ \boldsymbol{\Phi}(t,T_{k-1})\,\tilde{\boldsymbol{K}}_{k-1}(T_{k-1})\,\boldsymbol{\Phi}(t,T_{k-1})^T + \boldsymbol{\Phi}(t,T_k)\,\tilde{\boldsymbol{K}}_k(T_k)\,\boldsymbol{\Phi}(t,T_k)^T + \dots \\ &+ \tilde{\boldsymbol{K}}_p(T).\end{aligned} \tag{B.12}$$

Since $t < T_k$, the terms to the right of $\boldsymbol{\Phi}(t,T_k)\,\tilde{\boldsymbol{K}}_k(T_k)\,\boldsymbol{\Phi}(t,T_k)^T$ in (B.12) are equal to zero due to (B.11) and all instances of T_k are replaced with t as follows:

$$\begin{aligned}\boldsymbol{K}(t) = &\boldsymbol{\Phi}(t,0)\,\boldsymbol{K}_0\,\boldsymbol{\Phi}(t,0)^T + \boldsymbol{\Phi}(t,T_1)\,\tilde{\boldsymbol{K}}_1(T_1)\,\boldsymbol{\Phi}(t,T_1)^T + \dots \\ &+ \boldsymbol{\Phi}(t,T_{k-1})\,\tilde{\boldsymbol{K}}_{k-1}(T_{k-1})\,\boldsymbol{\Phi}(t,T_{k-1})^T + \boldsymbol{\Phi}(t,t)\,\tilde{\boldsymbol{K}}_k(t)\,\boldsymbol{\Phi}(t,t)^T \\ = &\boldsymbol{\Phi}(t,0)\,\boldsymbol{K}_0\,\boldsymbol{\Phi}(t,0)^T + \boldsymbol{\Phi}(t,T_1)\,\tilde{\boldsymbol{K}}_1(T_1)\,\boldsymbol{\Phi}(t,T_1)^T + \dots \\ &+ \boldsymbol{\Phi}(t,T_{k-1})\,\tilde{\boldsymbol{K}}_{k-1}(T_{k-1})\,\boldsymbol{\Phi}(t,T_{k-1})^T + \tilde{\boldsymbol{K}}_k(t).\end{aligned} \tag{B.13}$$

Clearly, the first term, middle terms, and last term on the right-hand side of (B.13) correspond respectively to the purely homogeneous solution with initial condition $\boldsymbol{K}_0$, other homogeneous solutions, and an inhomogeneous solution. Therefore, $\boldsymbol{K}(t)$ is shown to be obtainable by summation of all these solutions, which completes the proof.