Yunong Zhang · Dongsheng Guo

Zhang Functions and Various Models

Springer

Zhang Functions and Various Models

Yunong Zhang · Dongsheng Guo

Zhang Functions and Various Models

Springer

Yunong Zhang
School of Information Science
and Technology
Sun Yat-sen University
Guangzhou
China

Dongsheng Guo
School of Information Science
and Technology
Sun Yat-sen University
Guangzhou
China

ISBN 978-3-662-47333-7 ISBN 978-3-662-47334-4 (eBook)
DOI 10.1007/978-3-662-47334-4

Library of Congress Control Number: 2015940423

Springer Heidelberg New York Dordrecht London

Printed on acid-free paper

Springer-Verlag GmbH Berlin Heidelberg is part of Springer Science+Business Media
(www.springer.com)

To our ancestors and parents, as always

Preface

Time-varying mathematical problems are frequently encountered in scientific and engineering applications, such as circuit parameters in electronic circuits, aerodynamic coefficients in high-speed aircraft, and mechanical parameters in machinery. How to solve the time-varying problem effectively is becoming more and more necessary and important (as it is usually an essential part of many solutions). As we know, the common way to solve the time-varying problem is to treat such a problem as a static problem within a small time period (i.e., assume the short-time invariance of the problem). Then, the related numerical algorithms and/or neural-dynamics methods are developed to solve the problem at each single time instant, where the change trend of the time-varying coefficient(s) is not exploited. As for these conventional approaches, the computation is based on the present data, and the computed result is directly used for future. Thus, there exist lagging-error phenomena, when they are directly exploited to solve time-varying problems. In other words, the aforementioned approaches, which are designed theoretically/intrinsically for solving the static (or say, time-invariant, constant) problems, are less effective and efficient on time-varying problems solving.

Since March 12, 2001, Zhang et al. have formally proposed, investigated, and developed a special class of recurrent neural networks (i.e., Zhang neural network), which have been analyzed theoretically and substantiated comparatively for solving time-varying problems precisely and efficiently. By following the previous research on Zhang neural network, Zhang dynamics (ZD) has been generalized and further developed since 2008, whose state dimension can be multiple or one. It is viewed as a systematic approach to solving time-varying problems with the scalar situation included. It differs from conventional gradient dynamics (GD) in terms of the problem to be solved, error function, design formula, dynamic equation, and the utilization of time-derivative information. Besides, Zhang function (ZF), which is also referred to as Zhangian, is the design basis of ZD. It differs from the usual error/energy functions in the study of conventional approaches. Specifically, compared with the norm-based scalar-valued positive or at least lower-bounded energy function usually used in the GD design, ZF (1) is indefinite (i.e., can be positive, zero, or negative, in addition to being bounded, unbounded, or even lower unbounded), (2) can be matrix- or vector-valued (when solving a time-varying matrix- or vector-valued problem), and (3) can be real- or complex-valued (corresponding to a real- or complex-valued time-varying problem solving) to monitor and control the process of time-varying problems solving fully.

In this book, focusing on solving different types of time-varying problems, we design, propose, develop, analyze, model, and simulate various ZD models by defining various ZFs in real and complex domains. Specifically, in the real domain, we define three different classes of ZFs, i.e., scalar-valued ZFs, vector-valued ZFs and matrix-valued ZFs, for developing the resultant ZD models to solve the corresponding time-varying (scalar/vector/matrix-valued) problems. In the complex domain, we define different complex-valued ZFs for developing the resultant ZD models to solve three different types of complex-valued time-varying problems (one is with scalar formulation, and the rest are with matrix formulations). As for these ZD models, the related theoretical analyses are given, and the corresponding modeling (together with block diagrams) is illustrated. Computer simulations with various illustrative examples are performed to substantiate the efficacy of the proposed ZD models for time-varying problems solving. The simulation results also show the feasibility of the presented ZD approach (i.e., different ZFs leading to different ZD models) for real-time solution of time-varying problems. Based on these successful researches, we further apply such a ZD approach to repetitive motion planning (RMP) of redundant robot manipulators (including fixed-base and mobile ones). The corresponding results show the application prospect of the presented ZD approach to robots RMP.

The idea for this book on neural dynamics was conceived during classroom teaching as well as during research discussion in the laboratory and at international scientific meetings. Most of the materials in this book are derived from the authors' papers published in journals and proceedings of international conferences. In fact, since the early 1980s, the field of neural networks/dynamics has undergone phases of exponential growth, generating many new theoretical concepts and tools (including the authors' ones). At the same time, these theoretical results have been applied successfully to the solution of many practical problems. Our first priority is thus to cover each central topic in enough detail to make the material clear and coherent; in other words, each part (and even each chapter) is written in a relatively self-contained manner.

This book contains 15 chapters which are classified into the following five parts.

Part I: Scalar-Valued ZF in Real Domain (Chaps. 1–3);
Part II: Vector-Valued ZF in Real Domain (Chaps. 4–6);
Part III: Matrix-Valued ZF in Real Domain (Chaps. 7–10);
Part IV: ZF in Complex Domain (Chaps. 11–13);
Part V: ZF Application to Robot Control (Chaps. 14 and 15).

Chapter 1—In this chapter, we propose and develop four different indefinite ZFs as the error-monitoring functions, which lead to four different ZD models for time-varying reciprocal finding. In addition, theoretical analyses and Simulink modeling of such different ZD models are presented. Computer simulation results with three illustrative examples further substantiate the efficacy of the ZD models for time-varying reciprocal finding.

Chapter 2—In this chapter, by introducing six different ZFs, we propose, develop, and investigate six different ZD models to solve for time-varying inverse square root. In addition, this chapter presents theoretical analyses and Simulink modeling of such ZD models. Computer simulation results with two illustrative examples further substantiate the efficacy of the ZD models for time-varying inverse square root finding.

Chapter 3—In this chapter, six different ZD models are proposed, developed, and investigated by introducing six different ZFs for time-varying square root finding. In addition, the Simulink modeling of such ZD models is presented. Computer simulation results with two illustrative examples further substantiate the efficacy of the ZD models for time-varying square root finding.

Chapter 4—In this chapter, by following the idea of ZF, two ZD models are proposed, developed, and investigated for solving system of time-varying linear equations. In addition, it is theoretically proved that such two ZD models globally and exponentially converge to the theoretical time-varying solution of system of time-varying linear equations. Computer simulation results with three illustrative examples further substantiate the efficacy (as well as theoretical analyses) of the ZD models for solving system of time-varying linear equations.

Chapter 5—In this chapter, focusing on solving over-determined system of time-varying linear equations, we first propose, develop, and investigate two ZD models based on two different ZFs. Then, by introducing anther two different ZFs, another two ZD models are proposed, developed, and investigated for solving under-determined system of time-varying linear equations. Computer simulation results with four illustrative examples further substantiate the efficacy of such ZD models for solving over-determined and under-determined systems of time-varying linear equations.

Chapter 6—In this chapter, by introducing three different ZFs, we propose, develop, and investigate three different ZD models for solving time-varying linear matrix-vector inequality. Theoretical analyses and results are presented as well to show the excellent convergence performance of such ZD models. Computer simulation results with two illustrative examples further substantiate the efficacy of the ZD models for time-varying linear matrix-vector inequality solving.

Chapter 7—In this chapter, focusing on time-varying matrix inversion, we propose and develop six different ZFs that lead to six different ZD models. Meanwhile, a specific relationship between the ZD model and the Getz and Marsden (G-M) dynamic system is discovered. Eventually, theoretical analyses and Simulink modeling of such different ZD models are presented. Computer simulation results with two illustrative examples further substantiate the efficacy of the ZD models for time-varying matrix inversion.

Chapter 8—In this chapter, by introducing five different ZFs, we propose, develop, and investigate five different ZD models for time-varying matrix left pseudoinversion. In addition, the link between the ZD model and G-M dynamic system is discovered for time-varying matrix left pseudoinverse solving.

Theoretical analyses and computer simulation results with three illustrative examples further substantiate the efficacy of the ZD models on solving for the time-varying matrix left pseudoinverse.

Chapter 9—In this chapter, by introducing four different ZFs, four different ZD models are proposed, developed, and investigated for time-varying right pseudo-inversion. In addition, the link between the ZD model and G-M dynamic system is discovered to solve for time-varying matrix right pseudoinverse. Theoretical results and computer simulations with three illustrative examples further substantiate the efficacy of the ZD models for time-varying matrix right pseudoinversion.

Chapter 10—In this chapter, eight different indefinite ZFs, which lead to eight different ZD models, are proposed and developed as the error-monitoring functions for time-varying matrix square root finding. In addition, theoretical analyses and Simulink modeling of such ZD models are presented. Computer simulation results with two illustrative examples further substantiate the efficacy of the ZD models for time-varying matrix square root finding.

Chapter 11—In this chapter, by introducing four different ZFs in complex domain, four different ZD models are proposed, developed, and investigated to solve for time-varying complex reciprocal. Computer simulation results with three illustrative examples further substantiate the efficacy of the complex ZD models for time-varying complex reciprocal finding.

Chapter 12—In this chapter, focusing on time-varying complex matrix inversion, we propose, develop, and investigate three different complex ZD models by introducing three different complex ZFs. Computer simulation results with four illustrative examples further substantiate the efficacy of the complex ZD models for time-varying complex matrix inversion.

Chapter 13—In this chapter, by introducing five different complex ZFs, five different complex ZD models are proposed, developed, and investigated to solve for time-varying complex matrix generalized inverse (in most cases, the complex pseudoinverse). Meanwhile, theoretical analyses and results are presented to show the convergence properties of such complex ZD models. In addition, we discover the link between the complex ZD model and G-M dynamic system in complex domain. Computer simulation results with four illustrative examples further substantiate the efficacy of the complex ZD models for time-varying complex matrix generalized inverse solving.

Chapter 14—In this chapter, by introducing two different ZFs and by exploiting the ZD design formula, an acceleration-level RMP performance index is proposed, developed and investigated for fixed-base redundant robot manipulators. The resultant RMP scheme, which incorporates joint-angle, joint-velocity, and joint-acceleration limits, is further presented and investigated to remedy the joint-angle drift phenomenon of fixed-base redundant robot manipulators. Such a scheme is then reformulated as a quadratic program (QP), which is solved by a primal–dual neural network. With three path-tracking examples, computer simulation results based on PUMA560 robot manipulator substantiate well the effectiveness and accuracy of the acceleration-level RMP scheme, as well as show the application prospect of the presented ZD approach (i.e., different ZFs leading to different ZD models).

Chapter 15—In this chapter, by introducing three different ZFs and by exploiting the ZD design formula, we propose, develop, and investigate a velocity-level RMP performance index for mobile redundant robot manipulators. Then, based on such a performance and with physical limits considered, the resultant RMP scheme is presented and investigated to remedy the joint-angle drift phenomenon of mobile redundant robot manipulators. Such a scheme is reformulated as a QP, which is solved by a numerical algorithm. With two path-tracking examples, computer simulation results based on a wheeled mobile robot manipulator substantiate well the effectiveness and accuracy of the velocity-level RMP scheme, and show the application prospect of the presented ZD approach once again.

In summary, this book presents a novel approach (i.e., different ZFs resulting in different ZD models) for solving various time-varying problems in real and complex domains, and further applies such an approach to RMP control of different types of robot manipulators (showing its application prospect). This book is written for graduate students as well as academic and industrial researchers studying in the developing fields of neural dynamics, computer mathematics, time-varying computation, simulation and modeling, analog hardware, and robotics. It provides a comprehensive view of the combined research of these fields, in addition to its accomplishments, potentials, and perspectives. We do hope that this book will generate curiosity and also happiness to its readers for learning more in the fields and the research, and that it will provide new challenges to seek new theoretical tools and practical applications.

At the end of this Preface, it is worth pointing out that, in this book, a new and inspiring direction on the definition of the error function (or say, the energy function involved in convention researches) is provided for the neural-dynamics construction. This opens the door on defining the error function from a single definition equation of the specific problem to be solved to various appropriate formulations (resulting in various neural-dynamics models that can be chosen for practitioners in accordance with specific requests). It may promise to become a major inspiration for studies and researches in neural dynamics, time-varying problems solving, prediction, and dynamic decision making. Without doubt, this book can be extended. Any comments or suggestions are welcome. The authors can be contacted via e-mail: zhynong@mail.sysu.edu.cn, and gdongsh2008@126.com. The web page of Yunong Zhang is http://sist.sysu.edu.cn/~zhynong/.

Guangzhou, China
March 2015

Yunong Zhang
Dongsheng Guo

Acknowledgments

During the work on this book, we have had the pleasure of discussing its various aspects and results with many cooperators and students. We highly appreciate their contributions, which particularly allowed us to improve our book manuscript. Especially valuable help was provided by Dechao Chen, Liangyu He, Long Jin, Xiaogang Yan, and Zhengli Xiao. We are grateful to them for their help and suggestions.

The continuous support of our research by the National Natural Science Foundation of China (with number 61473323), by the Science and Technology Program of Guangzhou, China (with number 2014J4100057), by the Foundation of Key Laboratory of Autonomous Systems and Networked Control, Ministry of Education, China (with number 2013A07), and also by the National Innovation Training Program for University Students (with numbers 201410558065 and 201410558069) is gratefully acknowledged here either.

Besides, we would like to thank the editors sincerely for their very important and constructive comments and suggestions provided, in addition to their time and efforts spent in handling this book.

We are always very grateful to the nice people (especially the staff in Springer) for their strong support and encouragement during the preparation and publishing of this book.

Contents

Part I Scalar-Valued ZF in Real Domain

1 Time-Varying Reciprocal 3
1.1 Introduction and Preliminaries 3
1.2 ZFs and ZD Models 5
1.3 Theoretical Results and Analyses 8
1.4 Simulink Modeling 10
1.5 Illustrative Examples 11
1.6 Summary 15
References 16

2 Time-Varying Inverse Square Root 17
2.1 Introduction 17
2.2 ZFs and ZD Models 18
2.2.1 The First ZF and ZD Model 18
2.2.2 The Second ZF and ZD Model 19
2.2.3 The Third ZF and ZD Model 20
2.2.4 The Fourth ZF and ZD Model 20
2.2.5 The Fifth ZF and ZD Model 21
2.2.6 The Sixth ZF and ZD Model 21
2.3 Theoretical Results and Analyses 22
2.4 Simulink Modeling 24
2.5 Illustrative Examples 27
2.6 Summary 30
References 30

3 Time-Varying Square Root 33
3.1 Introduction 33
3.2 ZFs and ZD Models 34
3.3 Simulink Modeling 38

3.4 Illustrative Examples 39
3.5 Summary 44
References 45

Part II Vector-Valued ZF in Real Domain

4 System of Time-Varying Linear Equations 49
4.1 Introduction 49
4.2 ZFs and ZD Models 50
4.2.1 The First ZF and ZD Model 51
4.2.2 The Second ZF and ZD Model 51
4.3 Theoretical Results and Analyses 53
4.4 Illustrative Examples 55
4.5 Summary 59
References 59

5 Over-Determined and Under-Determined Systems of Time-Varying Linear Equations 61
5.1 Introduction 61
5.2 ZFs and ZD Models 62
5.2.1 With $m > n$ (Over-Determined System) 62
5.2.2 With $m < n$ (Under-Determined System) 63
5.3 Illustrative Examples 66
5.4 Summary 72
References 73

6 Time-Varying Linear Matrix-Vector Inequality 75
6.1 Introduction 75
6.2 ZFs and ZD Models 76
6.2.1 The First ZF and ZD Model 76
6.2.2 The Second ZF and ZD Model 77
6.2.3 The Third ZF and ZD Model 78
6.3 Illustrative Examples 82
6.4 Summary 87
References 87

Part III Matrix-Valued ZF in Real Domain

7 Time-Varying Matrix Inverse 91
7.1 Introduction 91
7.2 ZFs and ZD Models 92
7.3 Theoretical Results and Analyses 95

7.4 Simulink Modeling 97
7.5 Illustrative Examples 100
7.6 Summary 103
References 103

8 Time-Varying Matrix Left Pseudoinverse 105
8.1 Introduction 105
8.2 ZFs and ZD Models 107
8.2.1 The First ZF and ZD Model 107
8.2.2 The Second ZF and ZD Model 109
8.2.3 The Third ZF and ZD Model 109
8.2.4 The Fourth ZF and ZD Model 110
8.2.5 The Fifth ZF and ZD Model 111
8.3 Illustrative Examples 112
8.4 Summary 118
References 118

9 Time-Varying Matrix Right Pseudoinverse 121
9.1 Introduction 121
9.2 ZFs and ZD Models 122
9.3 Theoretical Results 124
9.4 Illustrative Examples 125
9.5 Summary 127
References 127

10 Time-Varying Matrix Square Root 129
10.1 Introduction 129
10.2 ZFs and ZD Models 130
10.3 Theoretical Results and Analyses 137
10.4 MATLAB Simulink Modeling 139
10.4.1 Simulink Blocks 139
10.4.2 Parameter Settings 140
10.5 Illustrative Examples 142
10.6 Summary 147
References 147

Part IV ZF in Complex Domain

11 Time-Varying Complex Reciprocal 151
11.1 Introduction 151
11.2 Complex ZFs and ZD Models 152

11.3 Illustrative Examples 156
11.4 Summary 160
References 160

12 Time-Varying Complex Matrix Inverse 163
12.1 Introduction 163
12.2 Complex ZFs and ZD Models 164
12.3 Illustrative Examples 166
12.4 Summary 171
References 171

13 Time-Varying Complex Matrix Generalized Inverse 173
13.1 Introduction 173
13.2 Complex ZFs and ZD Models 175
13.2.1 The First Complex ZD Model 176
13.2.2 The Second Complex ZD Model 179
13.2.3 The Third Complex ZD Model 180
13.2.4 The Fourth Complex ZD Model 181
13.2.5 The Fifth Complex ZD Model 182
13.3 Illustrative Examples 184
13.4 Summary 191
References 191

Part V ZF Application to Robot Control

14 Application to Fixed-Base Robot RMP 195
14.1 Introduction 195
14.2 RMP Performance Index Derived via Different ZFs 197
14.3 Scheme and QP Formulations 198
14.3.1 Acceleration-Level RMP Scheme 199
14.3.2 Bound Constraint Transformation Technique 199
14.3.3 QP Reformulation 200
14.3.4 QP Solver 200
14.4 Illustrative Examples 201
14.4.1 Pentagram-Path Tracking 202
14.4.2 East-Asian Character Writing 207
14.4.3 “V” Path Path Tracking 209
14.4.4 Comparisons with Velocity-Level RMP Scheme 210
14.5 Summary 212
References 212

15 Application to Mobile Robot RMP 215
15.1 Introduction 215
15.2 RMP Performance Index Derived via Different ZFs 217
15.2.1 Kinematics Modeling of Mobile Robot Manipulators 217
15.2.2 Velocity-Level RMP Performance Index 218
15.3 Scheme and QP Formulations 220
15.3.1 Velocity-Level RMP Scheme 220
15.3.2 QP Reformulation 222
15.3.3 QP Solver 223
15.4 Illustrative Examples 224
15.4.1 Circular Path Tracking 224
15.4.2 Lissajous-Figure Path Tracking 230
15.5 Summary 234
References 235

Acronyms

ASIC	Application-specific integrated circuit
DNN	Dual neural network
DOF	Degrees of freedom
FPGA	Field programmable gate array
GD	Gradient dynamics
MSSMRE	Maximal steady-state modeling residual error
PDNN	Primal–dual neural network
QP	Quadratic program
RMP	Repetitive motion planning
RT	Relative tolerance
VLSI	Very large-scale integration
ZD	Zhang dynamics
ZF	Zhang function

Part I
Scalar-Valued ZF in Real Domain

Chapter 1
Time-Varying Reciprocal

Abstract Along with neural dynamics (based on analog solvers) widely arising in scientific computation and optimization fields in recent decades which attracts extensive interest and investigation of researchers, a special type of neural dynamics, called Zhang dynamics (ZD), has been formally proposed by Zhang et al. for real-time solution of time-varying problems. By following Zhang et al.'s neural-dynamics design method, the ZD model, which is based on an indefinite Zhang function (ZF), can guarantee the exponential convergence performance for time-varying problems solving. In this chapter, for time-varying reciprocal finding, we propose, generalize, develop, and investigate different indefinite ZFs as the error-monitoring functions, which can lead to different ZD models. In addition, for the goal of developing the floating-point processors or coprocessors for the future generation of computers, the MATLAB Simulink modeling and simulative verifications of such different ZD models are presented. The modeling results further substantiate the efficacy of the proposed ZD models for time-varying reciprocal finding.

1.1 Introduction and Preliminaries

The reciprocal computation, which is described in the form of $f(x) = ax - 1 = 0$, is considered to be an important operation in a floating-point divider/processor. Thus, many researches on the reciprocal computation are conducted and presented [1–6]. However, these researches are just for the static reciprocal computation, thereby making the corresponding methods less accurate enough to solve the time-varying reciprocal problem in the following form:

$$f(x(t), t) = a(t)x(t) - 1 = 0 \in \mathbb{R},\ t \in [0, +\infty), \tag{1.1}$$

where $a(t) \neq 0 \in \mathbb{R}$ denotes a smoothly time-varying scalar with $\dot{a}(t) \in \mathbb{R}$ denoting the time derivative of $a(t)$, both of which are assumed to be known numerically or could be measured accurately. In this chapter, we aim at finding the $x(t) \in \mathbb{R}$ to make (1.1) hold true at any time instant $t \in [0, +\infty)$. Furthermore, $x^*(t)$ is used to denote the theoretical time-varying reciprocal of $a(t)$ [i.e., mathematically, $x^*(t) = 1/a(t)$].

Y. Zhang and D. Guo, *Zhang Functions and Various Models*,
DOI 10.1007/978-3-662-47334-4_1

Remark 1.1 The above $x^*(t)$ is given symbolically for better understanding and solution comparison, whose the computation of $1/a(t)$ at every single time instant t is less practical in real-life applications. Specifically, when we compute $1/a(t)$ at a time instant t, as the computation consumes time Δt inevitably, the value of $a(t)$ is changing during the computation procedure. Thus, the computed result is less accurate and less effective, since the value of the theoretical time-varying reciprocal has actually changed to $x^*(t+\Delta t)$ after the computation. This is the so-called lagging-error phenomenon. Note that, as for other time-varying problems solving (via the conventional computation approaches), such types of lagging-error phenomena still exist. This propels us to develop and investigate an effective computation approach for real-time solution of various time-varying problems (e.g., the ones presented and investigated in this chapter as well as Chaps. 2–13).

Generally speaking, in the solving process of (1.1), a real-time solver first receives the specific data of $a(t)$ at one single time instant; then the solver does computations based on the present and/or the stored previous data; and finally, it outputs the result to the user. Note that, in this process, the solver cannot use the future data because they are unknown and have not come yet at the present time instant. Furthermore, at every single time instant, we, based on the present and/or previous data, compute the result for future. This is also because computation consumes time inevitably. As for the conventional approaches, the computation is based on the present data, and the computed result is directly used for future. Thus, there exists the lagging-error problem (see also Remark 1.1), when they are directly exploited to solve the time-varying reciprocal problem. In other words, these approaches are less effective on the time-varying reciprocal problem solving. This is the reason why we need to develop and investigate an effective approach for time-varying reciprocal finding (and further, for real-time solution of various time-varying problems).

Being different from the conventional neural-dynamics approach (i.e., gradient dynamics, GD), a special type of neural dynamics, called Zhang dynamics (ZD), has been formally proposed by Zhang et al. for various time-varying problems solving [7–13]. According to Zhang et al.'s neural-dynamics design method, the ZD is designed based on an indefinite Zhang function (ZF) as the error-monitoring function (where the word "indefinite" here means that such an error-monitoring function can be positive, zero, negative or even lower-unbounded). This differs from the situation involved in the design of conventional approaches; for example, a norm-based positive-definite energy function is generally used in the GD design [8, 11, 12]. Thus, by making use of the time-derivative information of the time-varying coefficient(s) involved in the time-varying problem, the resultant ZD models can methodologically avoid the lagging errors generated by the conventional approaches. Note that such ZD models can guarantee much better convergence performance to the theoretical time-varying solution of the time-varying problem in an error-free manner. Besides, for better understanding and to lay a basis for further investigation, the concepts of ZD and ZF are presented as follows.

Concept 1.1 Zhang dynamics (ZD) has been generalized from Zhang neural network formally since 2008 [12], of which the state dimension can be multiple or one. It is viewed as a systematic approach to real-time solution of time-varying problems with scalar situation included as well. It differs from the conventional GD in terms of the problem to be solved, error function, design formula, dynamic equation, and the utilization of time-derivative information.

Concept 1.2 Zhang function (ZF), which is also referred to as Zhangian, is the design basis of ZD. It differs from the usual error/energy functions in the study of conventional approaches. Specifically, compared with the norm-based scalar-valued positive or at least lower-bounded energy function usually used in the GD design, ZF (1) is indefinite (i.e., can be positive, zero, or negative, in addition to being bounded, unbounded, or even lower unbounded), (2) can be matrix- or vector-valued (when solving a time-varying matrix- or vector-valued problem), and (3) can be real- or complex-valued (corresponding to a real- or complex-valued time-varying problem solving) to monitor and control the process of time-varying problems solving fully.

In this chapter, focusing on time-varying reciprocal finding, we propose, generalize, develop, and investigate different ZD models by defining different ZFs as the error-monitoring functions. In addition to the theoretical analyses and verifications of the convergence characteristics of the proposed ZD models, the MATLAB Simulink modeling [14–16] and illustrative examples are presented and investigated with the goal of developing the floating-point processors or coprocessors for the future generation of computers. From the modeling results, the efficacy of the proposed ZD models based on different ZFs for time-varying reciprocal finding is substantiated. To the best of the author's knowledge, almost all reported computation approaches [1–6] are theoretically/intrinsically designed for static/time-invariant reciprocal finding. There is almost no other literature handling such a specific problem solving, i.e., real-time solution of time-varying reciprocal, at present stage.

1.2 ZFs and ZD Models

In this section, we introduce four different ZFs and propose the resultant ZD models for solving the time-varying reciprocal problem (1.1).

Because the ZF is the design basis for deriving a ZD model and for presentation convenience, we denote the ZF by $e(t)$ with $\dot{e}(t)$ being the time derivative of $e(t)$. Note that, in this chapter and also in Chaps. 2 and 3, $e(t)$ and $\dot{e}(t)$ are used as the notations of the scalar-valued ZF and its time derivative, respectively. Besides, to lay a basis for further discussion, the design procedure for a ZD model is presented as follows.

- First, we define an indefinite ZF as the error-monitoring function to monitor the process of time-varying reciprocal finding.
- Second, to force $e(t)$ globally and exponentially converge to zero, we choose its time derivative $\dot{e}(t)$ via the following ZD design formula: [7–13]:

$$\dot{e}(t) = \frac{\mathrm{d}e(t)}{\mathrm{d}t} = -\gamma e(t), \tag{1.2}$$

where design parameter $\gamma > 0 \in \mathbb{R}$ corresponds to the reciprocal of a capacitance parameter, which should be set as large as the hardware would permit [10, 12, 13, 17], or selected appropriately for the simulative purpose.

- Finally, by expanding the ZD design formula (1.2), the dynamic equation of a ZD model is thus established for time-varying reciprocal finding.

For the excellent property of global and exponential convergence of the ZD design formula (1.2), we have the following theorem.

Theorem 1.1 *As for the ZD design formula (1.2) which is also a dynamic system, starting from an initial error $e(0) \in \mathbb{R}$, the error function $e(t) \in \mathbb{R}$ globally and exponentially converges to zero with rate γ.*

Proof For (1.2), by calculus, we obtain its analytical solution as $e(t) = e(0) \exp(-\gamma t)$. Based on the definition of global and exponential convergence, we can draw the conclusion that, starting from any $e(0)$, $e(t)$ globally and exponentially converges to zero with rate γ, as time t tends to infinity. The proof is thus complete. □

Besides, it is worth pointing out here that the aforementioned design procedure for the scalar situation can also be generalized for deriving the ZD models to solve other time-varying problems with matrix or vector formulations (e.g., the ones presented and investigated in Chaps. 4–10).

Specifically, for real-time solution of time-varying reciprocal problem (1.1), in this chapter, we define the following four different ZFs:

$$e(t) = x(t) - \frac{1}{a(t)}, \tag{1.3}$$

$$e(t) = a(t) - \frac{1}{x(t)}, \tag{1.4}$$

$$e(t) = a(t)x(t) - 1, \tag{1.5}$$

$$e(t) = \frac{1}{a(t)x(t)} - 1. \tag{1.6}$$

According to the ZD design formula (1.2), different ZFs lead to different ZD models, which is detailed as below.

- Let us consider the ZD design formula (1.2) and ZF (1.3). Then, we have

$$\dot{x}(t) + \frac{1}{a^2(t)}\dot{a}(t) = -\gamma\left(x(t) - \frac{1}{a(t)}\right),$$

which is rewritten as

$$a^2(t)\dot{x}(t) = -\dot{a}(t) - \gamma \left(a^2(t)x(t) - a(t)\right). \tag{1.7}$$

Thus, we obtain ZD model (1.7) for time-varying reciprocal finding.

- Considering the ZD design formula (1.2) and ZF (1.4), we have

$$\dot{a}(t) + \frac{1}{x^2(t)}\dot{x}(t) = -\gamma \left(a(t) - \frac{1}{x(t)}\right)$$

which is reformulated as

$$\dot{x}(t) = -\dot{a}(t)x^2(t) - \gamma \left(a(t)x^2(t) - x(t)\right). \tag{1.8}$$

Therefore, ZD model (1.8) for time-varying reciprocal finding is obtained.

- By combining the ZD design formula (1.2) and ZF (1.5), we have

$$\dot{a}(t)x(t) + a(t)\dot{x}(t) = -\gamma \left(a(t)x(t) - 1\right),$$

and then

$$a(t)\dot{x}(t) = -\dot{a}(t)x(t) - \gamma \left(a(t)x(t) - 1\right). \tag{1.9}$$

ZD model (1.9) for time-varying reciprocal finding is thus obtained.

- With the ZD design formula (1.2) and ZF (1.6) combined, we have

$$-\frac{1}{a^2(t)x^2(t)} \left(\dot{a}(t)x(t) + a(t)\dot{x}(t)\right) = -\gamma \left(\frac{1}{a(t)x(t)} - 1\right),$$

which is rewritten as

$$a(t)\dot{x}(t) = -\dot{a}(t)x(t) + \gamma \left(a(t)x(t) - a^2(t)x^2(t)\right). \tag{1.10}$$

Therefore, we come to ZD model (1.10) for time-varying reciprocal finding.

As a result, we have obtained four different types of ZD models [i.e. (1.7)–(1.10)] for time-varying reciprocal finding, which correspond to four different types of ZFs [i.e., (1.3)–(1.6)]. For readers' convenience and also for comparison, such four different ZD model based on four different ZFs for time-varying reciprocal finding are listed in Table 1.1.

Table 1.1 Different ZFs resulting in different ZD models for time-varying reciprocal finding

ZF	ZD model
(1.3)	$a^2(t)\dot{x}(t) = -\dot{a}(t) - \gamma\left(a^2(t)x(t) - a(t)\right)$
(1.4)	$\dot{x}(t) = -\dot{a}(t)x^2(t) - \gamma\left(a(t)x^2(t) - x(t)\right)$
(1.5)	$a(t)\dot{x}(t) = -\dot{a}(t)x(t) - \gamma\left(a(t)x(t) - 1\right)$
(1.6)	$a(t)\dot{x}(t) = -\dot{a}(t)x(t) + \gamma\left(a(t)x(t) - a^2(t)x^2(t)\right)$

1.3 Theoretical Results and Analyses

In this section, four propositions (viewed as the special cases of 1.1) are presented, which show the convergence properties of the proposed ZD models (1.7)–(1.10) for time-varying reciprocal finding.

Proposition 1.1 *Consider a smoothly time-varying scalar $a(t) \neq 0 \in \mathbb{R}$ involved in time-varying reciprocal problem (1.1). Starting from randomly-generated initial state $x(0) \neq 0 \in \mathbb{R}$ which has the same sign as $a(0)$, the neural state $x(t)$ of ZD model (1.7) derived from ZF (1.3) exponentially converges to the theoretical time-varying reciprocal $x^*(t)$ of $a(t)$ [i.e., $a^{-1}(t)$].*

Proof We use the well-known Lyapunov method to prove the exponential convergence of ZD model (1.7).

First, starting with ZF (1.3), we define a Lyapunov candidate

$$V(x(t), t) = \frac{1}{2}\left(x(t) - \frac{1}{a(t)}\right)^2 \geqslant 0,$$

where $V(x(t), t) = 0$ for any $x(t) = a^{-1}(t)$, and $V(x(t), t) > 0$ for any $x(t) \neq a^{-1}(t)$. Then, we derive its time derivative as

$$\begin{aligned}\dot{V}(x(t), t) = \frac{\mathrm{d}V(x(t), t)}{\mathrm{d}t} &= \left(x(t) - \frac{1}{a(t)}\right)\left(\dot{x}(t) + \frac{1}{a^2(t)}\dot{a}(t)\right)\\ &= -\gamma\left(x(t) - \frac{1}{a(t)}\right)^2 = -2\gamma V(x(t), t).\end{aligned}$$

Since $V(x(t), t) \geqslant 0$, then $\dot{V}(x(t), t) = -2\gamma V(x(t), t) \leqslant 0$, which guarantees the (final) negative-definiteness of $\dot{V}(x(t), t)$.

Furthermore, from $\dot{V}(x(t), t) = -2\gamma V(x(t), t)$, we have

$$V(x(t), t) = V(x(0), 0)\exp(-2\gamma t).$$

That is,

$$\frac{1}{2}\left(x(t)-\frac{1}{a(t)}\right)^2=\frac{1}{2}\left(x(0)-\frac{1}{a(0)}\right)^2\exp(-2\gamma t).$$

Thus, we have

$$\left|x(t)-\frac{1}{a(t)}\right|=\left|x(0)-\frac{1}{a(0)}\right|\exp(-\gamma t),$$

where symbol $|\cdot|$ denotes the absolute value of a scalar. With $\alpha=|x(0)-1/a(0)|$, the above equation is further rewritten as

$$\left|x(t)-\frac{1}{a(t)}\right|=\alpha\exp(-\gamma t),$$

which means that $x(t)$ exponentially converges to $a^{-1}(t)$ with the convergence rate $\gamma>0$. That is, starting from randomly-generated initial state $x(0)\neq 0\in\mathbb{R}$ which has the same sign as $a(0)$, the neural state $x(t)$ of ZD model (1.7) exponentially converges to the theoretical time-varying reciprocal $x^*(t)=a^{-1}(t)$ of $a(t)$ involved in time-varying Eq. (1.1). The proof is thus complete. $\square$

As for ZD models (1.8)–(1.10), we also have the following convergence results, with the related proofs being generalized from the proof of Proposition 1.1 (and being left to interested readers to complete as a topic of exercise).

Proposition 1.2 *Consider a smoothly time-varying scalar $a(t)\neq 0\in\mathbb{R}$ involved in time-varying reciprocal problem (1.1). Starting from randomly-generated initial state $x(0)\neq 0\in\mathbb{R}$ which has the same sign as $a(0)$, the neural state $x(t)$ of ZD model (1.8) derived from ZF (1.4) converges to the theoretical time-varying reciprocal $x^*(t)$ of $a(t)$ [i.e., $a^{-1}(t)$], with the error defined in ZF (1.4) exponentially convergent to zero.*

Proposition 1.3 *Consider a smoothly time-varying scalar $a(t)\neq 0\in\mathbb{R}$ involved in time-varying reciprocal problem (1.1). Starting from randomly-generated initial state $x(0)\neq 0\in\mathbb{R}$ which has the same sign as $a(0)$, the neural state $x(t)$ of ZD model (1.9) derived from ZF (1.5) converges to the theoretical time-varying reciprocal $x^*(t)$ of $a(t)$ [i.e., $a^{-1}(t)$], with the error defined in ZF (1.5) exponentially convergent to zero.*

Proposition 1.4 *Consider a smoothly time-varying scalar $a(t)\neq 0\in\mathbb{R}$ involved in time-varying reciprocal problem (1.1). Starting from randomly-generated initial state $x(0)\neq 0\in\mathbb{R}$ which has the same sign as $a(0)$, the neural state $x(t)$ of ZD model (1.10) derived from ZF (1.6) converges to the theoretical time-varying reciprocal $x^*(t)$ of $a(t)$ [i.e., $a^{-1}(t)$], with the error defined in ZF (1.6) exponentially convergent to zero.*

1.4 Simulink Modeling

For possible hardware implementation based on digital circuits and also for the goal of developing the floating-point processors or coprocessors for the future generation of computers, the MATLAB Simulink modeling of the proposed ZD models (1.7)–(1.10) is investigated and presented in this section. Before doing this, we need to transform some of such ZD models into the following explicit forms, with the corresponding block diagrams depicted in Fig. 1.1.

- For ZD model (1.7),

$$\dot{x}(t) = \left(1 - a^2(t)\right)\dot{x}(t) - \dot{a}(t) - \gamma\left(a^2(t)x(t) - a(t)\right).$$

- For ZD model (1.8), it is already in the explicit form, and does not need to be transformed.
- For ZD model (1.9),

$$\dot{x}(t) = (1 - a(t))\,\dot{x}(t) - \dot{a}(t)x(t) - \gamma\,(a(t)x(t) - 1).$$

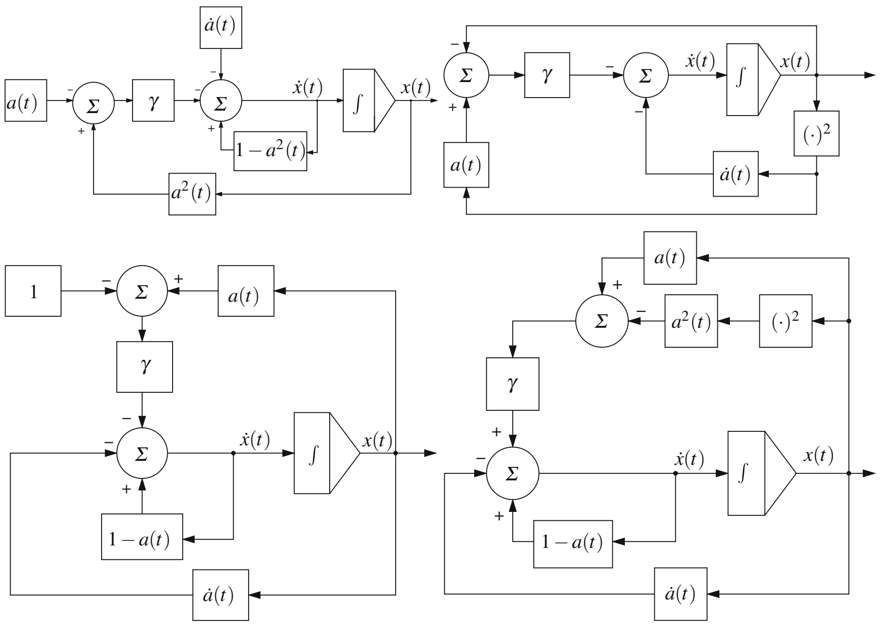


Fig. 1.1 Block diagrams of ZD models (1.7)–(1.10) for time-varying reciprocal finding

- For ZD model (1.10),

$$\dot{x}(t) = (1 - a(t))\,\dot{x}(t) - \dot{a}(t)x(t) + \gamma\left(a(t)x(t) - a^2(t)x^2(t)\right).$$

Therefore, the overall Simulink models of the proposed ZD models are shown in Figs. 1.2 and 1.3, in which $a(t)$ is generated by employing the "MATLAB Function" block using the "Clock" block as its input.

1.5 Illustrative Examples

In the previous sections, we have proposed the ZD models based on different ZFs for time-varying reciprocal finding, together with corresponding propositions and theoretical analyses. Based on the aforementioned Simulink models depicted in Figs. 1.2 and 1.3, the ensuing illustrative examples are shown to substantiate the efficacy of the proposed ZD models (1.7)–(1.10).

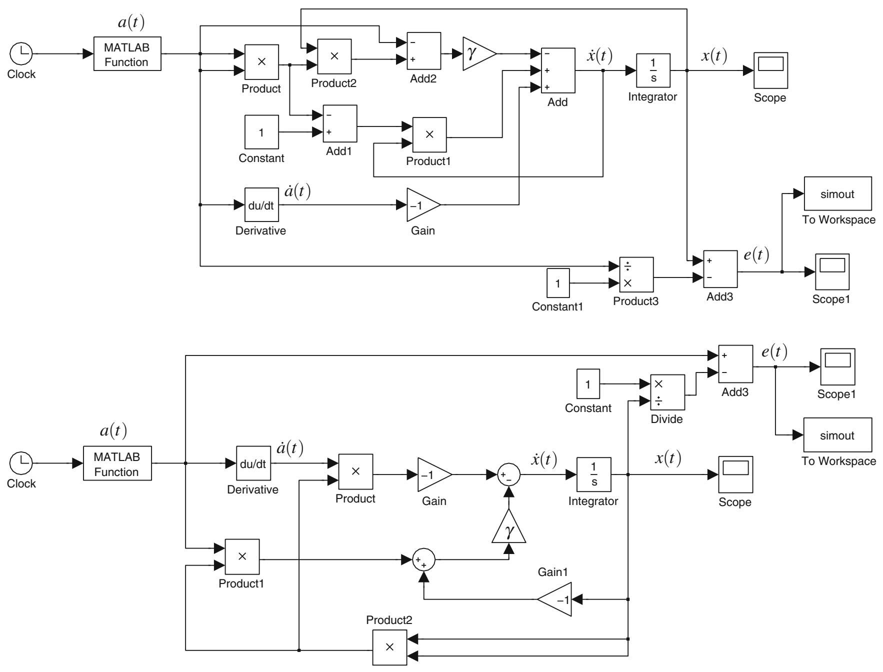


Fig. 1.2 Simulink modeling of ZD models (1.7) and (1.8) for time-varying reciprocal finding

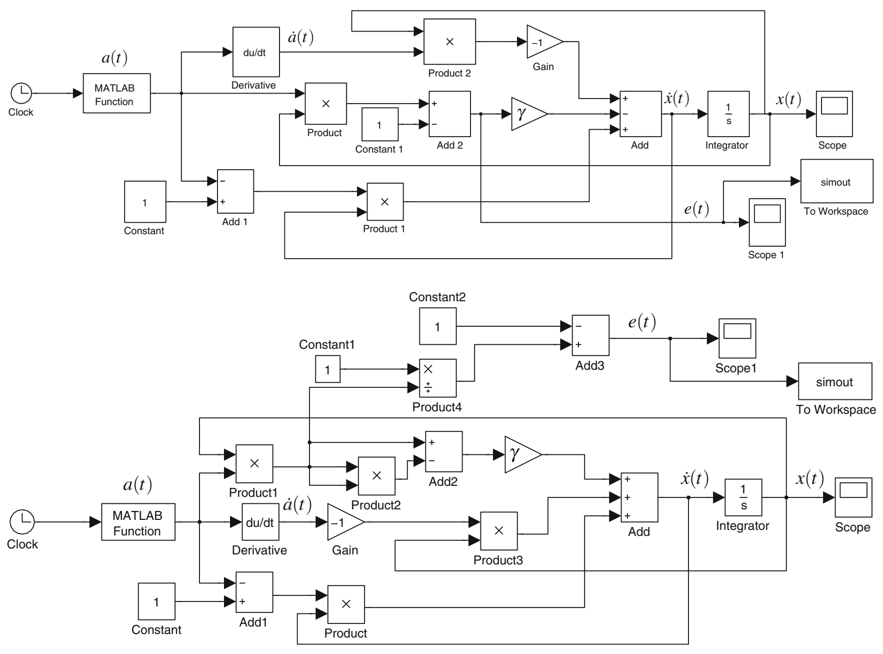


Fig. 1.3 Simulink modeling of ZD models (1.9) and (1.10) for time-varying reciprocal finding

Example 1.1 Let us consider the following time-varying reciprocal problem [in which $a(t) = \sin(3t) + \cos(\sin(2t)) + 2$ is involved]:

$$f(x(t), t) = (\sin(3t) + \cos(\sin(2t)) + 2)\, x(t) - 1 = 0. \tag{1.11}$$

The proposed ZD models (1.7)–(1.10) are exploited to solve this problem (1.11). It is easy to see that the theoretical initial reciprocal value is $x^*(0) = 1/a(0) \approx 0.333$. For convenience of observation, the initial state (or to say, starting value) $x(0)$ is randomly generated within [0.2, 0.4], i.e., $x(0) \in [0.2, 0.4]$. The simulation results based on the Simulink models are presented in Fig. 1.4. As shown in the figure, with design parameter $\gamma = 10$, the states $x(t)$ of the proposed ZD models all converge to the theoretical time-varying reciprocal $x^*(t) = a^{-1}(t)$, i.e., the theoretical time-varying solution of (1.11), in a rather short time (i.e., in less than 1 s). Through this example, we have primarily shown the efficacy of the proposed ZD models (1.7)–(1.10) for solving the time-varying reciprocal problem.

Example 1.2 In this example, we are considering a more complicated situation of the time-varying reciprocal problem, i.e.,

$$f(x(t), t) = (\sin(\cos(4t)) + \exp(-\sin(3t)) + \cos(t) + 2)\, x(t) - 1 = 0, \tag{1.12}$$

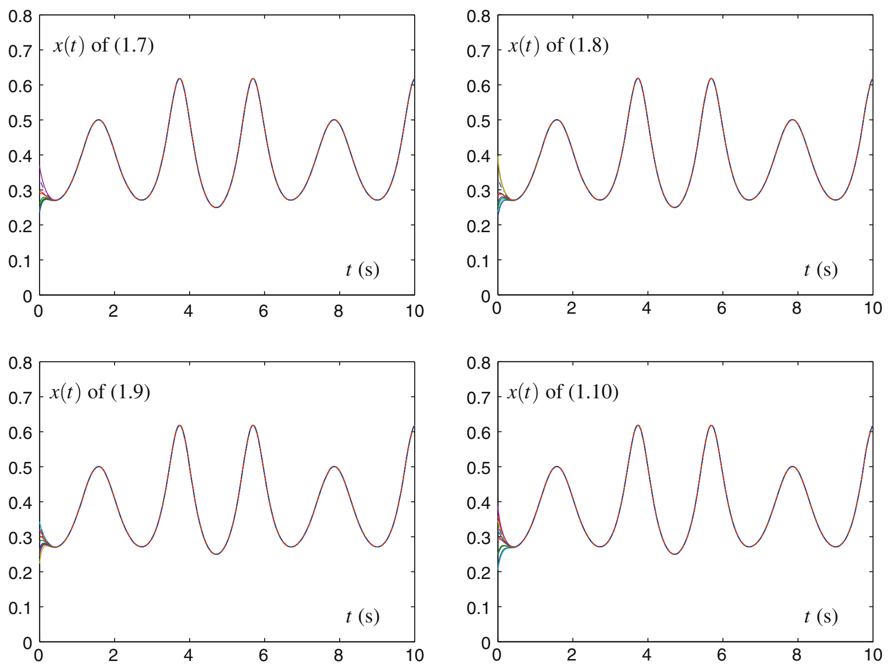


Fig. 1.4 State trajectories of ZD models (1.7)–(1.10) with design parameter $\gamma = 10$ for solving the time-varying reciprocal problem (1.11), where *dash-dotted curves* correspond to the theoretical time-varying reciprocal $a^{-1}(t)$, i.e., the theoretical solution of (1.11)

so as to investigate the general applicability of the proposed ZD models. In other words, $a(t) = \sin(\cos(4t)) + \exp(-\sin(3t)) + \cos(t) + 2$.

Similar to the way of Example 1.1, the proposed ZD models (1.7)–(1.10) are exploited to solve this reciprocal problem depicted in (1.12), and the corresponding simulation results are illustrated in Fig. 1.5. As shown in Fig. 1.5, with design-parameter $\gamma = 10$ and the initial state $x(0)$ randomly generated within $[0.1, 0.4]$, the state trajectories of the proposed ZD models (1.7)–(1.10) all fit well with the theoretical time-varying reciprocal $x^*(t)$, i.e., the theoretical time-varying solution of (1.12), rapidly (i.e., in less than 1 s).

From the above two examples, we can draw the conclusion that, either for a simple problem (1.11) or a more complicated problem (1.12), the proposed ZD models (1.7)–(1.10) based on different ZFs (1.3)–(1.6) can all solve the time-varying reciprocal problem efficiently. That is, with appropriate values of design parameter γ and initial state $x(0)$, the neural states of the proposed ZD models all converge to the time-varying theoretical solution of the time-varying reciprocal problem rapidly and accurately. Thus, the efficacy of the proposed ZD models (1.7)–(1.10) for time-varying reciprocal finding is substantiated well.

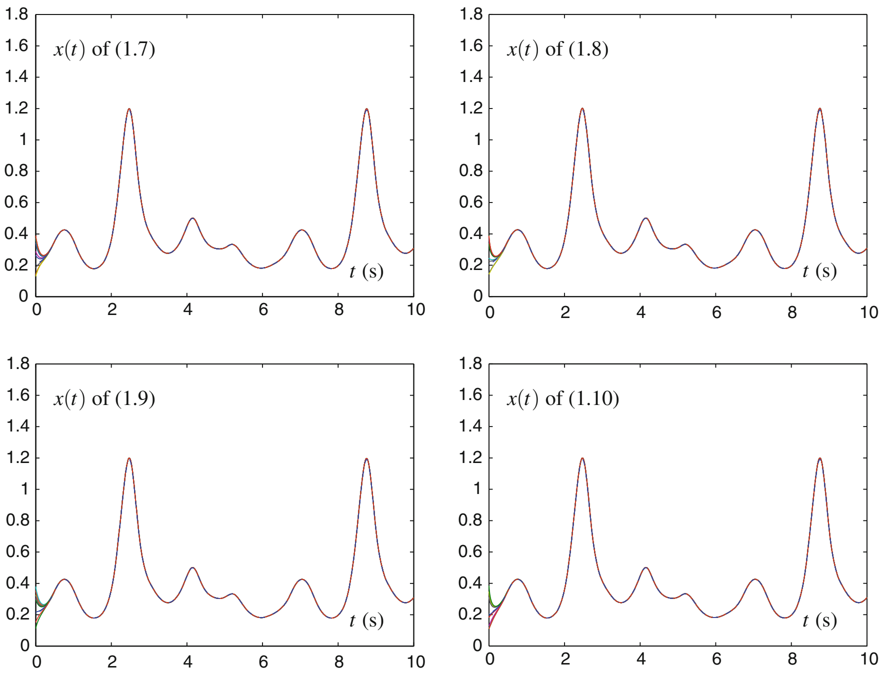


Fig. 1.5 State trajectories of ZD models (1.7)–(1.10) with design parameter $\gamma = 10$ for solving the time-varying reciprocal problem (1.12), where *dash-dotted curves* correspond to the theoretical time-varying reciprocal $a^{-1}(t)$, i.e., the theoretical solution of (1.12)

Example 1.3 As mentioned previously, the value of design parameter γ may affect the convergence performance of the proposed ZD models (1.7)–(1.10). Let us consider the following time-varying reciprocal problem [in which $a(t) = \sin(t)\cos(2t) + 2$]:

$$f(x(t), t) = (\sin(t)\cos(2t) + 2)\, x(t) - 1 = 0. \tag{1.13}$$

In this example, ZD model (1.7) is exploited to solve (1.13) with the initial state $x(0)$ randomly generated within [0.4, 0.6]. The computational errors $e(t) = x(t) - 1/(\sin(t)\cos(2t) + 2)$ with respect to different values of γ are displayed in Fig. 1.6. As seen from the figure, the convergence time of the computational error $e(t)$ to zero is becoming much shorter (i.e., from about 0.6 s to about 0.006) when the γ value increases from 10 to 1000. This simulation result indicates that design parameter γ plays an important role in the proposed ZD model (1.7), and should be set as large as the hardware would permit, or selected appropriately large for simulative purposes. Furthermore, the exponential-convergence characteristics of ZD model (1.7) can also be seen comparatively from Fig. 1.6. Note that the same conclusion applies to other

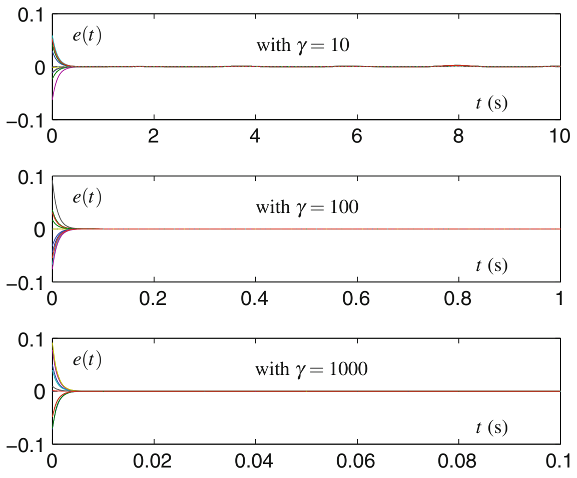


Fig. 1.6 Computational errors $e(t) = x(t) - 1/(\sin(t)\cos(2t) + 2)$ of ZD model (1.7) for solving the time-varying reciprocal problem (1.13) with different values of γ

ZD models [i.e., (1.8)–(1.10)], of which the related modeling results are omitted due to results similarity. Besides, the modeling testings of ZD models (1.8)–(1.10) are left to interested readers to complete as a topic of exercise.

1.6 Summary

In this chapter, by following Zhang et al.'s neural-dynamics design method and based on the Zhang function (ZF) as the error-monitoring function, a special type of neural dynamics, called Zhang dynamics (ZD), has been presented and investigated for real-time solution of time-varying reciprocal problem, which is in the time-varying form of $f(x(t), t) = a(t)x(t) - 1 = 0$ [i.e., (1.1)]. Specifically, based on different ZFs (1.3)–(1.6), different ZD models (1.7)–(1.10) have thus been proposed, generalized, developed, and investigated for time-varying reciprocal finding. Moreover, theoretical analyses have been given to substantiate the exponential convergence of the proposed ZD models. For possible hardware implementations based on digital circuits and for the goal of developing the floating-point processors or coprocessors for the future generation of computers, the MATLAB Simulink modeling of such proposed ZD models has been presented and investigated in this chapter. Through three illustrative examples, the efficacy of the proposed ZD models (1.7)–(1.10) has been further substantiated.

References

1. Pineiro J, Bruguera JD (2002) High-speed double-precision computation of reciprocal, division, squareroot, and inverse square root. IEEE Trans Comput 51(12):1377–1388
2. Hanrot G, Rivat J, Tenenbaum G, Zimmermann P (2003) Density results on floating-point invertible numbers. Theor Comput Sci 291(2):135–141
3. Kucukkabak U, Akkas A (2004) Design and implementation of reciprocal unit using table look-up and Newton-Raphson iteration. In: Proceedings of euromicro symposium on digital system design, pp 249–253
4. Croot E, Li R-C, Zhu HJ (2004) The *abc* conjecture and correctly rounded reciprocal square roots. Theor Comput Sci 315(2–3):405–417
5. Antelo E, Lang T, Montuschi P, Sannarelli A (2005) Low latency digit-recurrence reciprocal and square-root reciprocal algorithm and architecture. In: Proceedings of the 17th IEEE symposium on computer arithmetic, pp 147–154
6. Agrawal G, Khandelwal A, Swartzlander EE Jr (2007) An improved reciprocal approximation algorithm for a Newton Raphson divider. In: Proceedings of advanced signal processing algorithms, architectures, and implementations XVII, pp 1–12
7. Zhang Y, Jiang D, Wang J (2002) A recurrent neural network for solving Sylvester equation with time-varying coefficients. IEEE Trans Neural Netw 13(5):1053–1063
8. Zhang Y, Yi C, Guo D, Zheng J (2011) Comparison on Zhang neural dynamics and gradient-based neural dynamics for online solution of nonlinear time-varying equation. Neural Comput Appl 20(1):1–7
9. Zhang Y, Li Z (2009) Zhang neural network for online solution of time-varying convex quadratic program subject to time-varying linear-equality constrains. Phys Lett A 373(18–19):1639–1643
10. Zhang Y, Ma W, Cai B (2009) From Zhang neural network to Newton iteration for matrix inversion. IEEE Trans Circuits Syst I-Regul Pap 56(7):1405–1415
11. Zhang Y, Ge SS (2005) Design and analysis of a general recurrent neural network model for time-varying matrix inversion. IEEE Trans Neural Netw 16(6):1477–1490
12. Zhang Y, Yi C (2011) Zhang neural networks and neural-dynamic method. Nova Science Publishers, New York
13. Zhang Y, Li F, Yang Y, Li Z (2012) Different Zhang functions leading to different Zhang-dynamics models illustrated via time-varying reciprocal solving. Appl Math Model 36(9):4502–4511
14. Zhang Y, Guo X, Ma W (2008) Modeling and simulation of Zhang neural network for online time-varying equations solving based on MATLAB Simulink. In: Proceedings of the 7th international conference on machine learning and cybernetics, pp 805–810
15. Tan N, Chen K, Shi Y, Zhang Y (2009) Modeling, verification and comparison of Zhang neural net and gradient neural net for online solution of time-varying linear matrix equation. In: Proceedings of the 4th IEEE conference on industrial electronics and applications, pp 3698–3703
16. Ansari MS, Rahman SA (2011) DVCC-based non-linear feedback neural circuit for solving system of linear equations. Circuits, Syst Signal Process 30:1029–1045
17. Mead C (1989) Analog VLSI and neural systems. Addison-Wesley Longman, Boston

Chapter 2
Time-Varying Inverse Square Root

Abstract In this chapter, we propose, generalize, develop, and investigate different ZD models based on different ZFs for solving the time-varying inverse square root problem. In addition, this chapter shows modeling of the proposed ZD models using MATLAB Simulink techniques. The modeling results with different illustrative examples further substantiate the efficacy of such proposed ZD models for time-varying inverse square root finding.

2.1 Introduction

Inverse square root computation is an important numerical operation in many application areas such as digital signal processing, scientific computing, and computer graphics [1]. Specifically, inverse square root computation of a floating point scalar is exploited to compute a normalized vector [2], which can be used to determine lighting and reflection in a 3D graphics program [3]. The static inverse square root problem is generally formulated as $f(x) = ax^2 - 1$. Numerous numerical algorithms are investigated for solving such a problem [4–8]. Note that many potential computational algorithms are designed intrinsically for static (or to say, time-invariant, constant) problems solving [4] and associated with gradient methods [9]. As a result, almost all the previous algorithms are just effective for static inverse square root computation [10], and may not be accurate enough to solve the time-varying inverse square root problem [11–13].

In this chapter, focusing on time-varying inverse square root finding, we propose, generalize, develop, and investigate different ZD models by defining different ZFs as the error-monitoring functions. In addition, theoretical results are presented to show the convergence performance of such different ZD models. Moreover, MATLAB Simulink modeling [14–16] is shown for possible hardware realization of the proposed ZD models. Illustrative examples and modeling results further substantiate the efficacy of such proposed ZD models for time-varying inverse square root finding.

Y. Zhang and D. Guo, *Zhang Functions and Various Models*,
DOI 10.1007/978-3-662-47334-4_2

2.2 ZFs and ZD Models

In this section, different ZFs are introduced to construct different ZD models for time-varying inverse square root finding.

Let us consider the time-varying inverse square root problem, which is written in the form [11–13]

$$f(x(t), t) = a(t)x^2(t) - 1 = 0 \in \mathbb{R},\ t \in [0, +\infty), \tag{2.1}$$

where $a(t) > 0 \in \mathbb{R}$ denotes a smoothly time-varying scalar with $\dot{a}(t) \in \mathbb{R}$ denoting the time derivative of $a(t)$, both of which are assumed to be known numerically or could be measured accurately. In this chapter, we aim at finding the $x(t) \in \mathbb{R}$ at time instant $t \in [0, +\infty)$ to make (2.1) hold true. Furthermore, $x^*(t)$ is used to denote the theoretical time-varying inverse square root of $a(t)$ [i.e., mathematically, $x^*(t) = \pm 1/\sqrt{a(t)}$] in this chapter.

For solving the time-varying inverse square root problem (2.1), different ZD models based on different ZFs are thus developed and investigated, with the corresponding design procedures detailed as follows.

2.2.1 The First ZF and ZD Model

Following Zhang et al.'s design method [11–15, 17] (see also Sect. 1.2), we define the following indefinite ZF (i.e., the first ZF) as the error-monitoring function for time-varying inverse square root finding:

$$e(t) = x^2(t) - \frac{1}{a(t)}. \tag{2.2}$$

With ZF (2.2), by expanding the ZD design formula (1.2), we obtain

$$2x(t)\dot{x}(t) + \frac{1}{a^2(t)}\dot{a}(t) = -\gamma\left(x^2(t) - \frac{1}{a(t)}\right).$$

Hence, the ZD model based on ZF (2.2) for time-varying inverse square root finding is derived as follows:

$$\dot{x}(t) = -\frac{\dot{a}(t)}{2a^2(t)x(t)} - \frac{1}{2}\gamma\left(x(t) - \frac{1}{a(t)x(t)}\right). \tag{2.3}$$

Correspondingly, the block diagram of ZD model (2.3) is shown in the upper graph of Fig. 2.1.

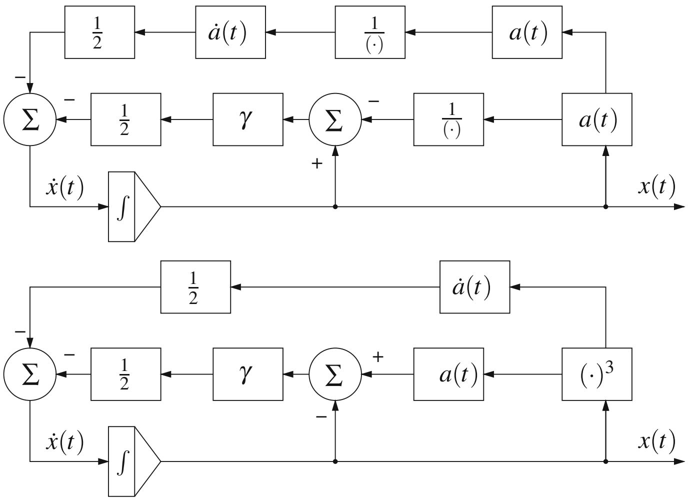


Fig. 2.1 Block diagrams of ZD (2.3) and (2.5) for time-varying inverse square root finding

2.2.2 The Second ZF and ZD Model

To introduce and show different ZD models based on different ZFs for solving the time-varying inverse square root problem (2.1), the second ZF is defined as

$$e(t) = a(t) - \frac{1}{x^2(t)}. \tag{2.4}$$

In view of ZF (2.4) and the ZD design formula (1.2), we obtain

$$\dot{a}(t) + \frac{2}{x^3(t)}\dot{x}(t) = -\gamma\left(a(t) - \frac{1}{x^2(t)}\right).$$

That is,

$$\dot{x}(t) = -\frac{1}{2}\dot{a}(t)x^3(t) - \frac{1}{2}\gamma\left(a(t)x^3(t) - x(t)\right), \tag{2.5}$$

which is the ZD model based on ZF (2.4) for time-varying inverse square root finding. Correspondingly, the block diagram of ZD model (2.5) is depicted in the lower graph of Fig. 2.1.

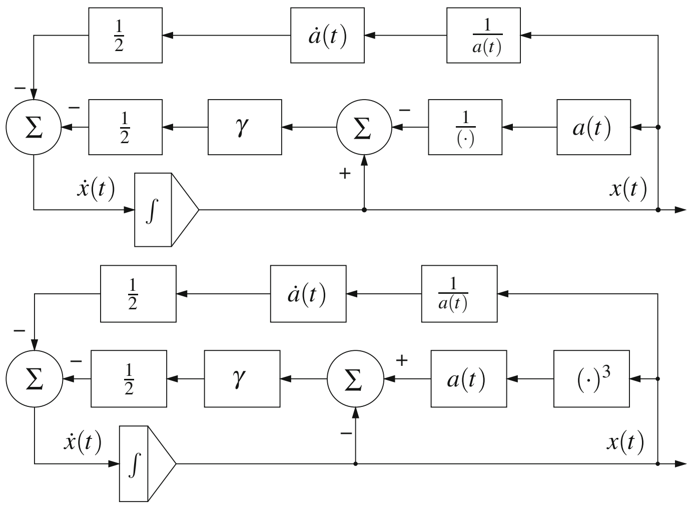


Fig. 2.2 Block diagrams of ZD (2.7) and (2.9) for time-varying inverse square root finding

2.2.3 The Third ZF and ZD Model

The first ZF (2.2) and the second ZF (2.4), which lead to ZD models (2.3) and (2.5), respectively, are defined above. Now, alternatively, we define the third ZF as

$$e(t) = a(t)x^2(t) - 1. \tag{2.6}$$

In view of the ZD design formula (1.2) and ZF (2.6), we have

$$\dot{a}(t)x^2(t) + 2a(t)x(t)\dot{x}(t) = -\gamma\left(a(t)x^2(t) - 1\right),$$

which is further written as

$$\dot{x}(t) = -\frac{\dot{a}(t)x(t)}{2a(t)} - \frac{\gamma}{2}\left(x(t) - \frac{1}{a(t)x(t)}\right). \tag{2.7}$$

Correspondingly, the block diagram of ZD model (2.7) for time-varying inverse square root finding is depicted in the upper graph of Fig. 2.2.

2.2.4 The Fourth ZF and ZD Model

For monitoring and controlling the process of the time-varying inverse square root problem (2.1) solving, another ZF (i.e., the fourth ZF) is defined in the following form:

$$e(t) = \frac{1}{a(t)x^2(t)} - 1. \tag{2.8}$$

With the ZD design formula (1.2) and ZF (2.8) considered, we have

$$-\frac{\dot{a}(t)x(t) + 2a(t)\dot{x}(t)}{a^2(t)x^3(t)} = -\gamma\left(\frac{1}{a(t)x^2(t)} - 1\right),$$

which yields the following differential equation of ZD:

$$\dot{x}(t) = -\frac{\dot{a}(t)x(t)}{2a(t)} - \frac{\gamma}{2}\left(a(t)x^3(t) - x(t)\right). \tag{2.9}$$

Correspondingly, the block diagram of ZD model (2.9) is shown in the lower graph of Fig. 2.2.

2.2.5 *The Fifth ZF and ZD Model*

Let us consider the following error-monitoring function (i.e., the fifth ZF):

$$e(t) = x(t) - \frac{1}{a(t)x(t)}. \tag{2.10}$$

As another form of ZF for time-varying inverse square root finding, ZF (2.10) leads to a different ZD model. Specifically, expanding the ZD design formula (1.2) with the aid of ZF (2.10), we obtain

$$\dot{x}(t) + \frac{\dot{a}(t)x(t) + a(t)\dot{x}(t)}{a^2(t)x^2(t)} = -\gamma\left(x(t) - \frac{1}{a(t)x(t)}\right),$$

which yields the following differential equation of ZD:

$$\dot{x}(t) = -\frac{\dot{a}(t)x(t)}{a^2(t)x^2(t) + a(t)} - \gamma\frac{a(t)x^3(t) - x(t)}{a(t)x^2(t) + 1}. \tag{2.11}$$

2.2.6 *The Sixth ZF and ZD Model*

The sixth and also the last ZF is given finally as

$$e(t) = a(t)x(t) - \frac{1}{x(t)}. \tag{2.12}$$

Table 2.1 Different ZFs resulting in different ZD models for finding time-varying inverse square root

ZF	ZD model
(2.2)	$\dot{x}(t) = -\frac{\dot{a}(t)}{2a^2(t)x(t)} - \frac{1}{2}\gamma\left(x(t) - \frac{1}{a(t)x(t)}\right)$
(2.4)	$\dot{x}(t) = -\frac{1}{2}\dot{a}(t)x^3(t) - \frac{1}{2}\gamma\left(a(t)x^3(t) - x(t)\right)$
(2.6)	$\dot{x}(t) = -\frac{\dot{a}(t)x(t)}{2a(t)} - \frac{1}{2}\gamma\left(x(t) - \frac{1}{a(t)x(t)}\right)$
(2.8)	$\dot{x}(t) = -\frac{\dot{a}(t)x(t)}{2a(t)} - \frac{1}{2}\gamma\left(a(t)x^3(t) - x(t)\right)$
(2.10)	$\dot{x}(t) = -\frac{\dot{a}(t)x(t)}{a^2(t)x^2(t)+a(t)} - \gamma\frac{a(t)x^3(t)-x(t)}{a(t)x^2(t)+1}$
(2.12)	$\dot{x}(t) = -\frac{\dot{a}(t)x^3(t)}{a(t)x^2(t)+1} - \gamma\frac{a(t)x^3(t)-x(t)}{a(t)x^2(t)+1}$

Substituting ZF (2.12) into the ZD design formula (1.2), we have

$$\dot{a}(t)x(t) + a(t)\dot{x}(t) + \frac{\dot{x}(t)}{x^2(t)} = -\gamma\left(a(t)x(t) - \frac{1}{x(t)}\right),$$

which yields the following differential equation of ZD:

$$\dot{x}(t) = -\frac{\dot{a}(t)x^3(t)}{a(t)x^2(t)+1} - \gamma\frac{a(t)x^3(t)-x(t)}{a(t)x^2(t)+1}. \tag{2.13}$$

Due to similarity to the block diagrams of the previous ZD models, the block diagrams of ZD models (2.11) and (2.13) are omitted.

In summary, we have obtained six different ZD models [i.e., (2.3), (2.5), (2.7), (2.9), (2.11) and (2.13)] for solving the time-varying inverse square root problem (2.1). Such six ZD models are derived from six different ZFs (2.2), (2.4), (2.6), (2.8), (2.10) and (2.12), respectively. For comparison purposes and reading convenience, the proposed different ZD models based on different ZFs are listed in Table 2.1.

2.3 Theoretical Results and Analyses

In this section, theoretical results and analyses are presented, which show the convergence properties of the proposed ZD models (2.3), (2.5), (2.7), (2.9), (2.11) and (2.13) for finding time-varying inverse square root.

Proposition 2.1 *Consider a smoothly time-varying positive scalar $a(t) > 0 \in \mathbb{R}$ involved in (2.1). Starting from any initial state $x(0) \neq 0 \in \mathbb{R}$, we have*

- *if $x(0) > 0$, the neural state $x(t) \in \mathbb{R}$ of ZD model (2.3) based on ZF (2.2) converges to the positive theoretical time-varying inverse square root of $a(t)$, with ZF (2.2) exponentially converging to zero; and,*
- *if $x(0) < 0$, the neural state $x(t) \in \mathbb{R}$ of ZD model (2.3) based on ZF (2.2) converges to the negative theoretical time-varying inverse square root of $a(t)$, with ZF (2.2) exponentially converging to zero.*

Proof We use the well-known Lyapunov theory to prove the convergence performance of ZD model (2.3).

Let us first define the following Lyapunov function candidate, which is clearly nonnegative:

$$V(x(t),t) = \frac{1}{2}\left(x^2(t) - \frac{1}{a(t)}\right)^2 \geqslant 0,$$

where $V(x(t),t) = 0$ only if $x(t) = x^*(t) = \pm 1/\sqrt{a(t)}$ and $V(x(t),t) > 0$ if $x(t) \neq x^*(t) = \pm 1/\sqrt{a(t)}$. In addition, if $e(t) = x^2(t) - 1/a(t)$ tends to infinity [correspondingly $|x(t) - x^*(t)| \to \infty$], we have $V(x(t),t) \to \infty$ as well.

Then, along the state trajectory of ZD model (2.3), we derive the time derivative of $V(x(t),t)$ as

$$\begin{aligned}\dot{V}(x(t),t) = \frac{\mathrm{d}V(x(t),t)}{\mathrm{d}t} &= \left(x^2(t) - \frac{1}{a(t)}\right)\left(2x(t)\dot{x}(t) + \frac{1}{a^2(t)}\dot{a}(t)\right)\\ &= -\gamma\left(x^2(t) - \frac{1}{a(t)}\right)^2 = -2\gamma V(x(t),t) \leqslant 0,\end{aligned}$$

which guarantees the final negative-definiteness of $\dot{V}(x(t),t)$.

Thus, in accordance with the Lyapunov theory, $x(t)$ converges to $x^*(t)$. In addition, as seen from the dynamic equation of ZD model (2.3), zero cannot be a divisor, i.e., state $x(t) \neq 0$. In view of the solution continuity of ZD model (2.3), we obtain the following results. If initial state $x(0) > 0$, the neural state $x(t) \in \mathbb{R}$ of ZD model (2.3) converges to the positive theoretical time-varying inverse square root of $a(t)$, i.e., $x(t) \to x^*(t) = 1/\sqrt{a(t)}$. Otherwise [i.e., if initial state $x(0) < 0$], the neural state $x(t) \in \mathbb{R}$ of ZD model (2.3) converges to the negative theoretical time-varying inverse square root of $a(t)$, i.e., $x(t) \to x^*(t) = -1/\sqrt{a(t)}$.

Furthermore, from $\dot{V}(x(t),t) = -2\gamma V(x(t),t)$, we have

$$V(x(t),t) = V(x(0),0)\exp(-2\gamma t).$$

That is,

$$\frac{1}{2}\left(x^2(t) - \frac{1}{a(t)}\right)^2 = \frac{1}{2}\left(x^2(0) - \frac{1}{a(0)}\right)^2 \exp(-2\gamma t).$$

Thus, we have

$$\left|x^2(t) - \frac{1}{a(t)}\right| = \left|x^2(0) - \frac{1}{a(0)}\right| \exp(-\gamma t).$$

By setting $\alpha = |x^2(0) - 1/a(0)|$, the above equation becomes

$$\left|x^2(t) - \frac{1}{a(t)}\right| = \alpha \exp(-\gamma t),$$

which indicates that $x^2(t)$ exponentially converges to $1/a(t)$, i.e., ZF (2.2) exponentially converges to zero. Therefore, the proof is complete. □

As for the other five ZD models, we also have the corresponding convergence results. Concerning the convergence performance of the proposed ZD models based on six different ZFs, the following unified proposition is presented, with the related proof being generalized from the proof of Proposition 2.1 and being left to interested readers to complete as a topic of exercise.

Proposition 2.2 *Consider a smoothly time-varying positive scalar $a(t) > 0 \in \mathbb{R}$ involved in (2.1). Starting from any initial state $x(0) \neq 0 \in \mathbb{R}$,*

- *if $x(0) > 0$, the neural states $x(t) \in \mathbb{R}^+$ of ZD models listed in Table 2.1 converge to the positive theoretical time-varying inverse square root of $a(t)$, with ZFs listed in Table 2.1 exponentially converging to zero, respectively; and,*
- *if $x(0) < 0$, the neural states $x(t) \in \mathbb{R}^-$ of ZD models listed in Table 2.1 converge to the negative theoretical time-varying inverse square root of $a(t)$, with ZFs listed in Table 2.1 exponentially converging to zero, respectively.*

2.4 Simulink Modeling

In this section, the MATLAB Simulink modeling of the proposed ZD models [i.e., (2.3), (2.5), (2.7), (2.9), (2.11) and (2.13)] is investigated and presented. The overall Simulink models of such proposed ZD models for time-varying inverse square root finding are shown in Figs. 2.3 and 2.4.

Some important parameters and options related to the Simulink models which we establish in Figs. 2.3 and 2.4, etc., are specified as follows.

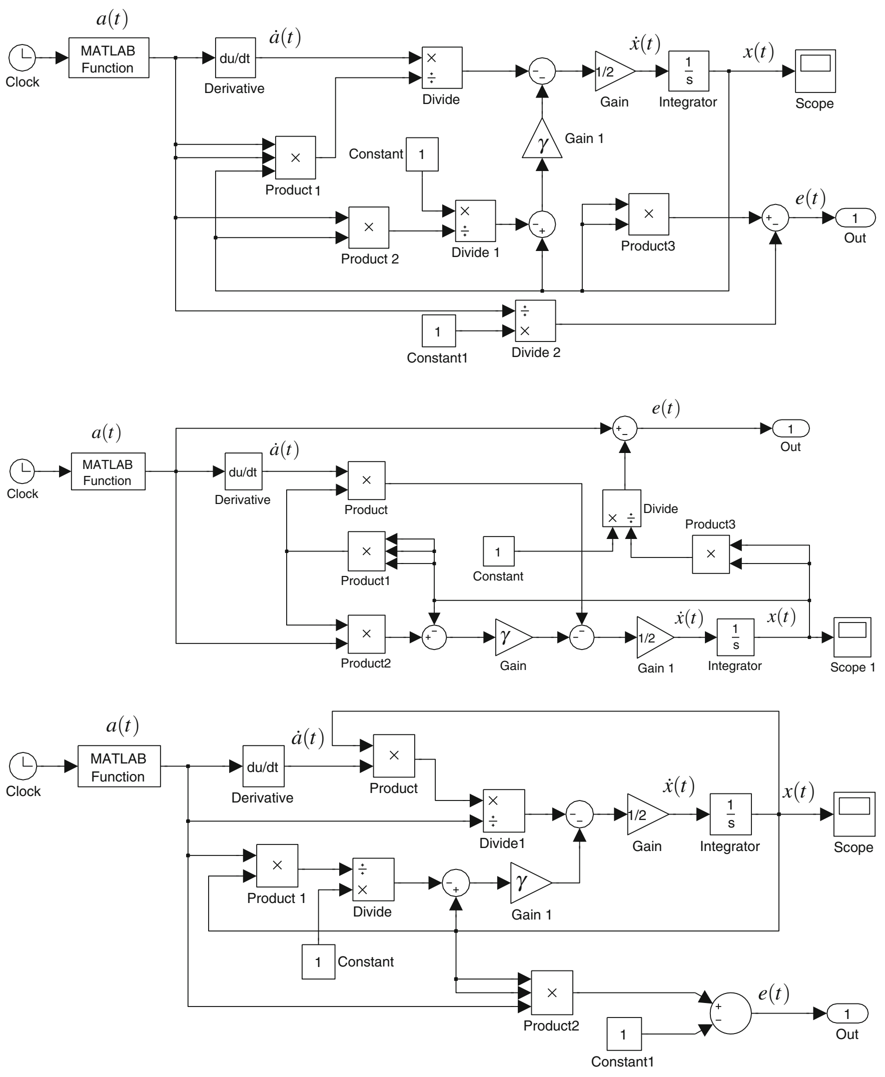


Fig. 2.3 Modeling of ZD (2.3), (2.5) and (2.7) for finding time-varying inverse square root

- $a(t)$ is generated by employing the "MATLAB Function" block with the "Clock" block as its input.
- Open the "Configuration Parameters" dialog box and set the options "Solver" to be "ode15s", "Max step size" to be "0.1", and "Min step size" to be "auto". In addition, by default, the option "Relative tolerance" is set as "1e-5" (i.e., 10^{-5}), and "Absolute tolerance" is "auto".

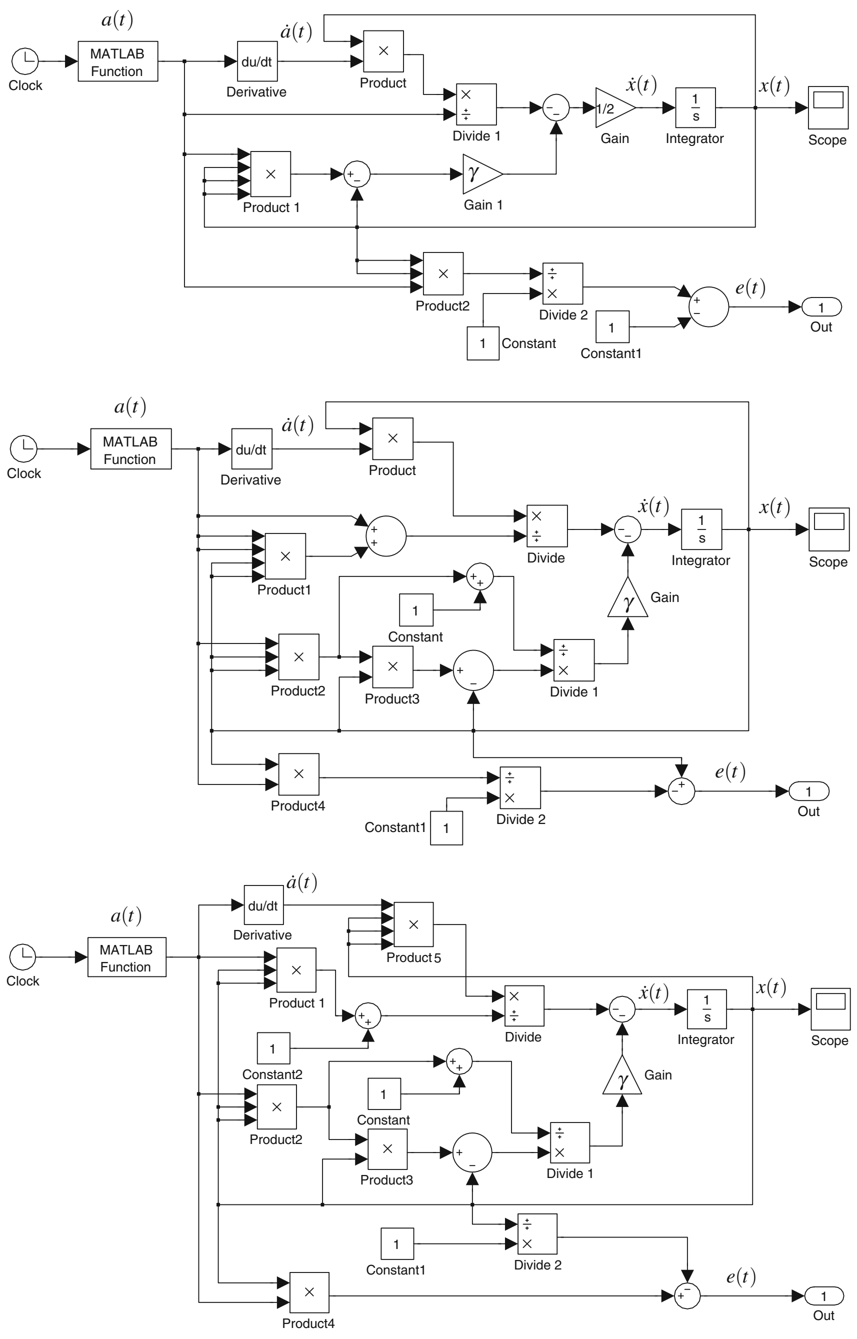


Fig. 2.4 Modeling of ZD (2.9), (2.11) and (2.13) for time-varying inverse square root finding

2.5 Illustrative Examples

In this section, based on the MATLAB Simulink modeling techniques, we substantiate the convergence performance of the proposed ZD models through two examples.

Example 2.1 Let us consider the time-varying inverse square root problem (2.1) with $a(t) = 8\cos(\sin(5t)) + 3\sin(2t) + 1$; i.e.,

$$f(x(t), t) = (8\cos(\sin(5t)) + 3\sin(2t) + 1)\, x^2(t) - 1 = 0. \tag{2.14}$$

The proposed ZD models (2.3), (2.5), (2.7), (2.9), (2.11) and (2.13) with $\gamma = 10$ are exploited to solve the above problem (2.14). Note that the theoretical initial solutions of (2.14) are

$$x^*(0) = \pm\frac{1}{\sqrt{8\cos(\sin 0) + 3\sin 0 + 1}} \approx \pm 0.33.$$

For convenience of observation, we randomly generate the positive and negative initial states $x(0)$, respectively, within $[0.2, 0.5]$ and $[-0.5, -0.2]$, which are the intervals around the theoretical initial solutions ± 0.33. As shown in Fig. 2.5, the neural states of ZD models [denoted by solid curves] converge to the theoretical time-varying inverse square root $x^*(t)$ [denoted by dash–dotted curves] of (2.14) in a short time (about 0.5 s). Therefore, we have substantiated the efficacy of the proposed ZD models for solving the time-varying inverse square root problem with an appropriate value of γ. It is worth pointing out that, even if we set the initial states $x(0) \neq 0$ far away from the theoretical initial solutions ± 0.33, the neural states $x(t)$ of the proposed ZD models (2.3), (2.5), (2.7), (2.9), (2.11) and (2.13) also converge to the theoretical time-varying solutions of (2.14) rapidly.

Example 2.2 In this example, we discuss how the value of design parameter γ affects the convergence performance of the proposed ZD models.

First, let us exploit ZD models (2.3) and (2.7) to solve the following time-varying inverse square root problem with $\gamma = 10$, 100 and 1000, respectively:

$$f(x(t), t) = (2\sin(3t) + 3\exp(\cos(2t)) + 7)\, x^2(t) - 1 = 0. \tag{2.15}$$

Starting with ten randomly-generated initial states $x(0)$, five of which are in $[0.2, 0.3]$ and the others are in $[-0.3, -0.2]$, we have the modeling results of computational error $e(t)$ shown in Fig. 2.6. As seen from the upper graph of Fig. 2.6, the maximal steady-state modeling error of ZD model (2.3) becomes much smaller when the value of design parameter γ increases. Specifically speaking, when $\gamma = 10$, the order of the maximal steady-state modeling error is 10^{-3}; and, when $\gamma = 1000$, the order of the maximal steady-state modeling error decreases to be 10^{-5}. This means that the maximal steady-state modeling error of ZD model (2.3) decreases in an $O(\gamma^{-1})$ manner. Note that the integrator precision (i.e., the relative tolerance of the integrator

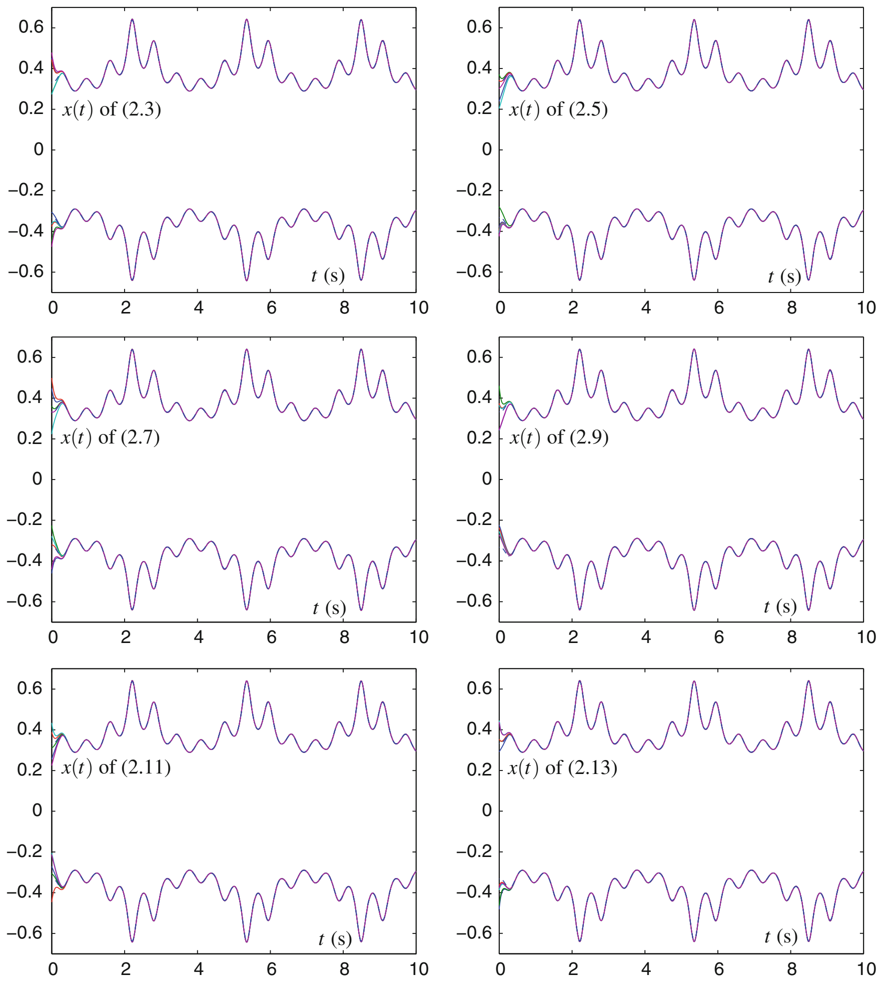


Fig. 2.5 State trajectories of ZD models (2.3), (2.5), (2.7), (2.9), (2.11) and (2.13) with $\gamma = 10$ for solving the time-varying inverse square root problem (2.14)

denoted as RT in this chapter) in the Simulink models also influences the convergence performance and accuracy of the ZD models.

In addition, in the previous modeling results, we set the integrator precision to be 1.0×10^{-5}. Figure 2.7 shows that, when we increase the integrator precision (i.e., decrease RT), the maximal steady-state modeling error of ZD model (2.3) is decreased as well. Theoretically speaking, the maximal steady-state modeling error can decrease to zero, when the relative tolerance RT tends to zero.

Moreover, the convergence speed of the proposed ZD models for solving (2.15) can be expedited effectively by increasing γ. This is shown in the upper and lower graphs of Fig. 2.6 with ZD models (2.3) and (2.7), respectively, as examples. Thus,

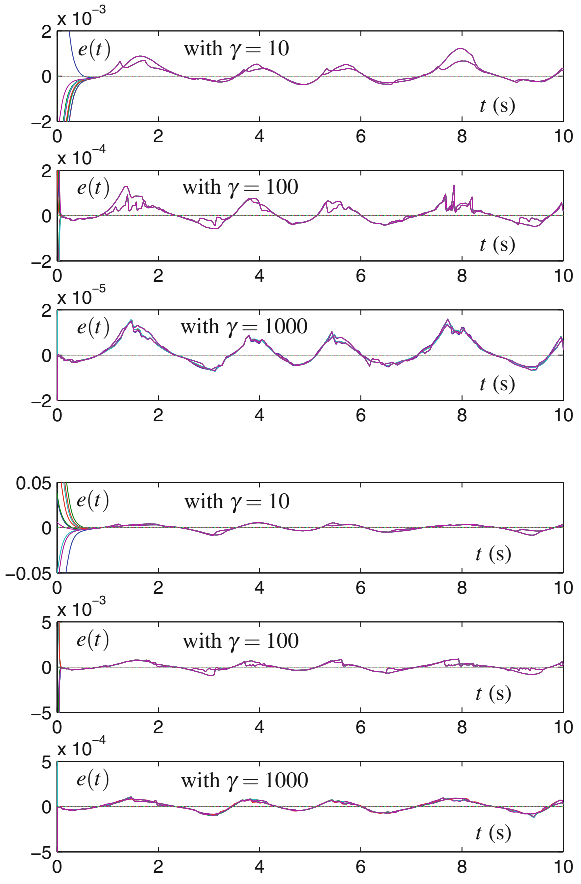


Fig. 2.6 Modeling errors $e(t)$ of ZD models (2.3) and (2.7) with the integrator's relative tolerance being 10^{-5} and with different γ for solving the time-varying inverse square root problem (2.15)

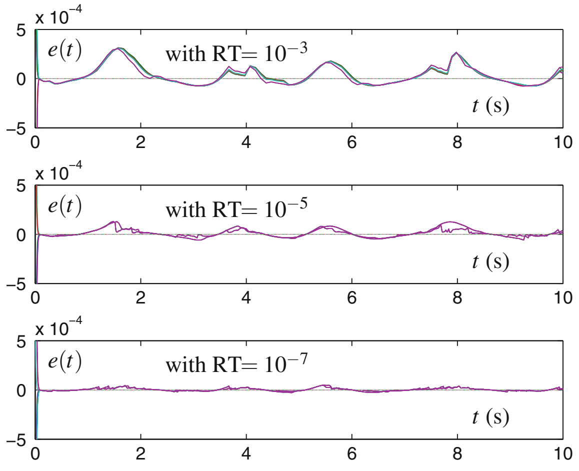


Fig. 2.7 Modeling errors $e(t)$ of ZD model (2.3) with $\gamma = 100$ and with different values of relative tolerance (RT) for solving the time-varying inverse square root problem (2.15)

we should set the value of γ appropriately large not only to decrease the modeling error, but also to shorten the convergence time.

In summary, we have the conclusion that the design parameter γ plays an important role in ZD models (2.3) and (2.7) on the convergence performance (including engineering accuracy). It is worth pointing out that the same conclusion applies to other ZD models [i.e., (2.5), (2.9), (2.11) and (2.13)], whose related modeling results are omitted due to similarity of results. Besides, the modeling tests of ZD models (2.5), (2.9), (2.11) and (2.13) are left to interested readers to complete as a topic of exercise.

2.6 Summary

In this chapter, six different ZD models based on six different ZFs have been proposed, generalized, developed, and investigated for solving the time-varying inverse square root problem in the form of $f(x(t), t) = a(t)x^2(t) - 1 = 0$. In addition, theoretical results and analyses have been given to substantiate the exponential convergence of the proposed ZD models. For possible hardware implementation, the MATLAB Simulink modeling of the proposed ZD models has also been presented and investigated in this chapter. The illustrative modeling results have further substantiated the efficacy of the proposed ZD models on time-varying inverse square root finding.

References

1. Seidel PM (1999) High-speed redundant reciprocal approximation. Integr VLSI J 28(1):1–12
2. Blinn J (2003) Jim Blinn's corner: notation, notation, notation. Elsevier, San Francisco
3. Eberly D (2001) 3D game engine design. Elsevier, San Francisco
4. Clenshaw CW, Olver FWJ (1986) Unrestricted algorithms for reciprocals and square roots. BIT Numer Math 26(4):475–492
5. Lang T, Montuschi P (1999) Very high radix square root with prescaling and rounding and a combined division/square root unit. IEEE Trans Comput 48(8):827–841
6. Pineiro JA, Bruguera JD (2002) High-speed double-precision computation of reciprocal, division, square root, and inverse square root. IEEE Trans Comput 51(12):1377–1388
7. Ercegovac MD, Lang T, Muller JM, Tisserand A (2000) Reciprocation, square root, inverse square root, and some elementary functions using small multipliers. IEEE Trans Comput 49(7):628–637
8. Mead C (1989) Analog VLSI and neural systems. Addison-Wesley Longman, Boston
9. Zhang Y, Yi C, Guo D, Zheng J (2011) Comparison on Zhang neural dynamics and gradient-based neural dynamics for online solution of nonlinear time-varying equation. Neural Comput Appl 20(1):1–7
10. Zhang Y, Leithead WE, Leith DJ (2005) Time-series Gaussian process regression based on Toeplitz computation of $O(N^2)$ operations and $O(N)$-level storage. In: Proceedings of the 44th IEEE international conference on decision and control, pp 3711–3716
11. Zhang Y, Li Z, Xie Y, Tan H, Chen P (2013) Z-type and G-type ZISR (Zhang inverse square root) solving. In: Proceedings of the 4th international conference on intelligent control and information processing, pp 123–128

12. Zhang Y, Li Z, Guo D, Li W, Chen P (2013) Z-type and G-type models for time-varying inverse square root (TVISR) solving. Soft Comput 17(11):2021–2032
13. Zhang Y, Yu X, Xie Y, Tan H, Fan Z (2013) Solving for time-varying inverse square root by different ZD models based on different Zhang functions. In: Proceedings of the 25th Chinese control and decision conference, pp 1358–1363
14. Zhang Y, Guo X, Ma W (2008) Modeling and simulation of Zhang neural network for online time-varying equations solving based on MATLAB Simulink. In: Proceedings of the 7th international conference on machine learning and cybernetics, pp 805–810
15. Tan N, Chen K, Shi Y, Zhang Y (2009) Modeling, verification and comparison of Zhang neural net and gradient neural net for online solution of time-varying linear matrix equation. In: Proceedings of the 4th IEEE conference on industrial electronics and applications, pp 3698–3703
16. Ansari MS, Rahman SA (2011) DVCC-based non-linear feedback neural circuit for solving system of linear equations. Circuits Syst Signal Process 30(5):1029–1045
17. Zhang Y, Yi C (2011) Zhang neural networks and neural-dynamic method. Nova Science Publishers, New York

Chapter 3
Time-Varying Square Root

Abstract In this chapter, focusing on time-varying square root finding, we propose, generalize, develop, and investigate different ZFs as the error-monitoring functions, which lead to different ZD models. Then, toward the final purpose of field programmable gate array (FPGA) and application-specific integrated circuit (ASIC) realization, the MATLAB Simulink modeling and verification of such different ZD models are shown. Both theoretical analysis and modeling results further substantiate the efficacy of the proposed ZD models for time-varying square root finding.

3.1 Introduction

The problem of solving time-varying square root in the form of $x^2(t) - a(t) = 0$ and/or static (or termed, constant, time-invariant) square root in the form of $x^2 - a = 0$ is considered to be a basic mathematical operation arising in science, commercial computer, and engineering fields. Any system (e.g., the hardware units and software packages) complying with the IEEE 754 binary floating-point standard [1], including the majority of current microprocessors, contains the square root implementation in the operation set. These systems must support and evaluate this operation accurately and correctly. Thus, many numerical algorithms have been developed and investigated for square root finding [2–8]. However, it may not be efficient enough for most numerical algorithms due to their serial-processing nature performed on digital computers [9]. Suitable for analog VLSI implementation [10, 11] and in view of potential high-speed parallel processing, the neural-dynamics approach has now been regarded as a powerful alternative to online problem solving [12–15]. Besides, it is worth mentioning that most reported computational-schemes are theoretically/intrinsically designed for solving time-invariant (or termed, static, constant) problems (i.e., time-invariant square root finding) and/or related to those used for division [16, 17].

In this chapter, we propose, generalize, develop, and investigate six continuous-time ZD models for solving the time-varying square root problem by defining six different ZFs as the error-monitoring functions and constructing six first-order differential equations to force the corresponding ZFs converge to zero. Then, the MATLAB

Y. Zhang and D. Guo, *Zhang Functions and Various Models*,
DOI 10.1007/978-3-662-47334-4_3

Simulink modeling [18] and verification are presented with the final purpose of field programmable gate array (FPGA) and application-specific integrated circuit (ASIC) realization [19]. The modeling results with different illustrative examples further substantiate the efficacy of the proposed ZD models based on different ZFs for time-varying square root finding.

3.2 ZFs and ZD Models

In this section, we introduce six different ZFs and propose the resultant ZD models for time-varying square root finding (together with the convergence results of such different ZD models).

Let us consider the following time-varying square root problem:

$$x^2(t) - a(t) = 0 \in \mathbb{R},\ \ t \in [0, +\infty), \tag{3.1}$$

where $a(t) > 0 \in \mathbb{R}$ denotes a smoothly time-varying scalar, which, together with time derivative $\dot{a}(t) \in \mathbb{R}$, is assumed to be known numerically or can be measured accurately. Our objective is to find $x(t) \in \mathbb{R}$ in real time t such that the above smoothly time-varying nonlinear equation (3.1) holds true. For presentation convenience, in this chapter, let $x^*(t) \in \mathbb{R}$ denote the theoretical time-varying square root of $a(t)$ [i.e., mathematically, $x^*(t) = \pm\sqrt{a(t)}$].

For solving the above time-varying square root problem (3.1), we define six different ZFs as below:

$$e(t) = \frac{1}{a(t)}x^2(t) - 1, \tag{3.2}$$

$$e(t) = \frac{1}{x^2(t)}a(t) - 1, \tag{3.3}$$

$$e(t) = x^2(t) - a(t), \tag{3.4}$$

$$e(t) = \frac{1}{x^2(t)} - \frac{1}{a(t)}, \tag{3.5}$$

$$e(t) = x(t) - a(t)\frac{1}{x(t)}, \tag{3.6}$$

$$e(t) = \frac{1}{a(t)}x(t) - \frac{1}{x(t)}. \tag{3.7}$$

According to Zhang et al.'s design method [12–15] presented in Sect. 1.2, with the ZD design formula (1.2), different ZFs lead to different ZD models for time-varying square root finding. Therefore, we have different ZD models as follows.

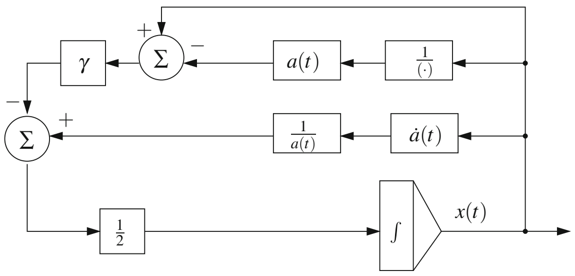


Fig. 3.1 Block diagram of ZD model (3.8) for time-varying square root finding

- Considering the ZD design formula (1.2) and ZF (3.2), we have

$$-\frac{\dot{a}(t)}{a^2(t)}x^2(t)+\frac{2}{a(t)}x(t)\dot{x}(t)=-\gamma\left(\frac{1}{a(t)}x^2(t)-1\right).$$

  Thus, the following dynamic equation (i.e., a first-order differential equation) of a ZD model is obtained for time-varying square root finding:

$$\dot{x}(t)=\frac{\dot{a}(t)x(t)}{2a(t)}-\frac{\gamma}{2}\left(x(t)-\frac{a(t)}{x(t)}\right). \tag{3.8}$$

  In order to display the ZD model (3.8) better, the resultant block diagram of such a ZD model is shown in Fig. 3.1.
- Considering the ZD design formula (1.2) and ZF (3.3), we have

$$\dot{x}(t)=\frac{\dot{a}(t)x(t)}{2a(t)}-\frac{\gamma}{2}\left(\frac{x^3(t)}{a(t)}-x(t)\right). \tag{3.9}$$

  Based on ZF (3.3), another ZD model [i.e., (3.9)] is obtained for time-varying square root finding. Besides, Fig. 3.2 shows the resultant block diagram of such a ZD model.
- Considering the ZD design formula (1.2) and ZF (3.4), the following ZD model is established for time-varying square root finding:

$$\dot{x}(t)=\frac{\dot{a}(t)-\gamma\left(x^2(t)-a(t)\right)}{2x(t)}. \tag{3.10}$$

  The resultant block diagram of ZD model (3.10) is shown in Fig. 3.3.

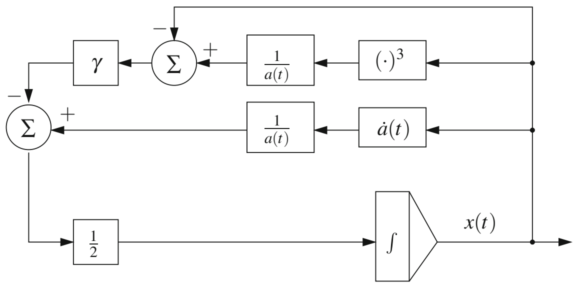


Fig. 3.2 Block diagram of ZD model (3.9) for time-varying square root finding

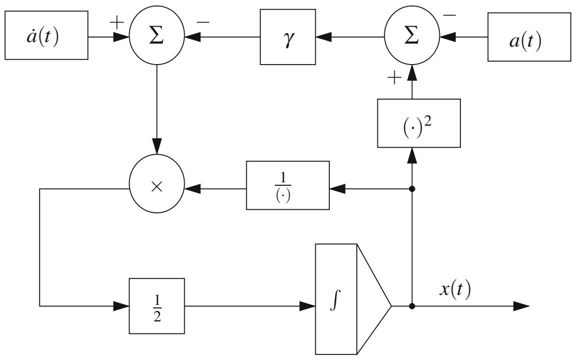


Fig. 3.3 Block diagram of ZD model (3.10) for time-varying square root finding

- With (1.2) and (3.5) exploited, we have the following ZD model for time-varying square root finding:

$$\dot{x}(t) = \frac{\dot{a}(t)x^3(t)}{2a^2(t)} - \frac{\gamma}{2}\left(\frac{x^3(t)}{a(t)} - x(t)\right). \tag{3.11}$$

  Besides, Fig. 3.4 shows the block diagram of ZD model (3.11).
- With (1.2) and (3.6) exploited, we have the following ZD model for time-varying square root finding:

$$\dot{x}(t) = \frac{\dot{a}(t)x(t)}{x^2(t) + a(t)} - \gamma\frac{x^3(t) - a(t)x(t)}{x^2(t) + a(t)}. \tag{3.12}$$

- Based on the ZD design formula (1.2) and ZF (3.7), we have

$$\dot{x}(t) = \frac{\dot{a}(t)x^3(t)}{a(t)x^2(t) + a^2(t)} - \gamma\frac{x^3(t) - a(t)x(t)}{x^2(t) + a(t)}. \tag{3.13}$$

  Thus, we obtain ZD model (3.13) based on ZF (3.7) for time-varying square root finding. Note that the block diagrams of ZD models (3.12) and (3.13) are omitted for the similarity, and are left to interested readers to complete as a topic of exercise.

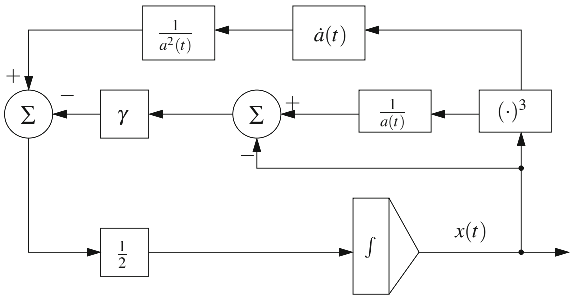


Fig. 3.4 Block diagram of ZD model (3.11) for time-varying square root finding

In summary, we have obtained six different ZD models [i.e., (3.8)–(3.13)] for time-varying square root finding, which correspond to six different ZFs [i.e., (3.2)–(3.7)]. For readers' convenience, such six different ZD models corresponding to six different ZFs are listed in Table 3.1.

Besides, as for the convergence property of the proposed ZD models (3.8)–(3.13), we have the following proposition. Note that the proof of such a proposition can be generalized from that of Theorem 1.1 or 2.1 presented in the previous chapters (and the related proof is also left to interested readers to complete as a topic of exercise).

Proposition 3.1 *Consider a smooth time-varying scalar $a(t) \in \mathbb{R}$ involved in the nonlinear equation (3.1), which is positive at $t \in [0, +\infty)$. For each of ZD models (3.8)–(3.13), starting from randomly-generated positive (or negative) initial state $x(0) \neq 0 \in \mathbb{R}$, the corresponding ZF exponentially converges to zero [which implies that neural-state $x(t) \in \mathbb{R}$ of the ZD model converges to the theoretical positive (or negative) time-varying square root $x^*(t)$ of $a(t)$].*

Remark 3.1 Based on the above analysis, all the resultant ZD models are effective on time-varying square root finding, though they are based on different ZFs (i.e.,

Table 3.1 Different ZFs and ZD models for time-varying inverse square root finding

ZF	ZD model
(3.2)	$\dot{x}(t) = \frac{\dot{a}(t)x(t)}{2a(t)} - \frac{\gamma}{2}\left(x(t) - \frac{a(t)}{x(t)}\right)$
(3.3)	$\dot{x}(t) = \frac{\dot{a}(t)x(t)}{2a(t)} - \frac{\gamma}{2}\left(\frac{x^3(t)}{a(t)} - x(t)\right)$
(3.4)	$\dot{x}(t) = \frac{\dot{a}(t) - \gamma\left(x^2(t) - a(t)\right)}{2x(t)}$
(3.5)	$\dot{x}(t) = \frac{\dot{a}(t)x^3(t)}{2a^2(t)} - \frac{\gamma}{2}\left(\frac{x^3(t)}{a(t)} - x(t)\right)$
(3.6)	$\dot{x}(t) = \frac{\dot{a}(t)x(t)}{x^2(t) + a(t)} - \gamma\frac{x^3(t) - a(t)x(t)}{x^2(t) + a(t)}$
(3.7)	$\dot{x}(t) = \frac{\dot{a}(t)x^3(t)}{a(t)x^2(t) + a^2(t)} - \gamma\frac{x^3(t) - a(t)x(t)}{x^2(t) + a(t)}$

the error-monitoring functions). Note that the results presented in this chapter as well as the previous chapters (i.e., Chaps. 1 and 2) give us a new direction on the definition of the error function for the neural-dynamics construction. Specifically, the error function is not only defined by following directly the definition equation of the problem to be solved [e.g., the Eqs. (1.1), (2.1) and (3.1) involved in Chaps. 1, 2 and this chapter respectively], but also defined as other appropriate forms. Evidently, this opens the door on defining the error function from a single definition equation (of the problem) to various appropriate formulations. More importantly, different neural-dynamics models are thus obtained by defining different error functions for the problem to be solved (e.g., different ZD models based on different ZFs in this book), which may be an inspiring direction on the research of neural dynamics.

3.3 Simulink Modeling

According to the proposed ZD models (3.8)–(3.13) and the block diagrams shown in Figs. 3.1, 3.2, 3.3, and 3.4, the corresponding MATLAB Simulink modelings of such ZD models are investigated and presented in this section for possible circuits implementation and also for the final purpose of FPGA and ASIC realization. As a graphical design based modeling tool, MATLAB Simulink exploits existing function blocks to construct mathematical and logical models as well as process flow. The Simulink modeling can be viewed as a virtual implementation of a real system satisfying a set of requirements. Moreover, the dynamic model developed in MATLAB Simulink environment can be extended to the HDL (Hardware Description Language) code and then to the final FPGA and ASIC realization [19].

The overall Simulink modeling of the proposed ZD models for time-varying square root finding is illustrated in Figs. 3.5 and 3.6. Note that, for the overall Simulink modelings shown in Figs. 3.5 and 3.6, some of the default modeling-environment options should be changed. The options setting can be done by using the "Configuration Parameters" dialog box in the MATLAB Simulink environment, with some important parameter settings specified as follows.

- Starting time (e.g., 0.0) and Stop time (e.g., 10.0);
- Solver (i.e., integrator algorithm): "ode45 (Dormand-Prince)";
- Max step size: "0.2" and Min step size: "auto";
- Initial step size: "auto";
- Relative tolerance: "1e-6" (i.e., 10^{-6});
- Absolute tolerance: "auto".

In addition, the check box in front of "States" of the option "Data Import/Export" should be selected, which is for the purpose of better displaying the ZD-modeling results and is associated with the "StopFcn" code (of "Callbacks" in the dialog box entitled "Model Properties" which is started from the "File" pull-down menu). This is shown in Fig. 3.7.

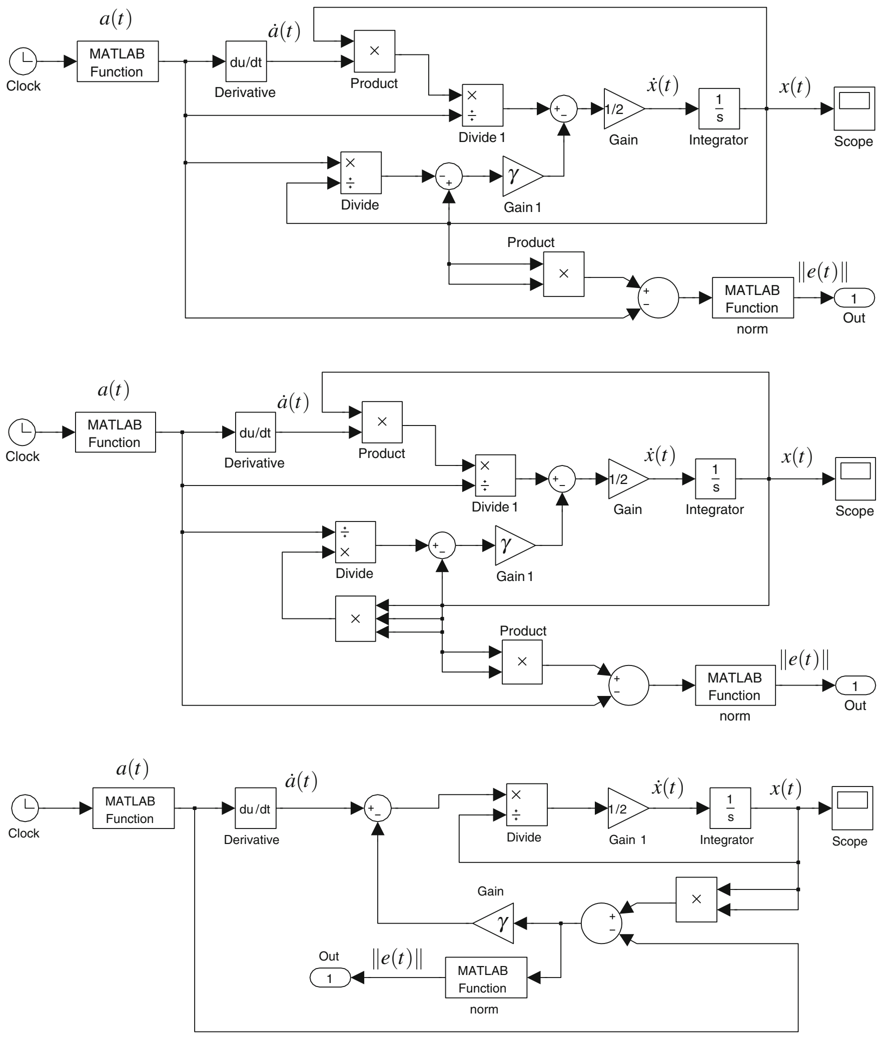


Fig. 3.5 Simulink modeling of ZD models (3.8)–(3.10) for time-varying square root finding

3.4 Illustrative Examples

Based on the overall Simulink modeling depicted in Figs. 3.5 and 3.6, the ensuing illustrative examples are presented and investigated to substantiate the efficacy of the proposed ZD models (3.8)–(3.13) for time-varying square root finding.

Example 3.1 Let us consider the time-varying square root problem (3.1) with $a(t) = 2\cos(5t) + 3\sin(\cos(2t)) + 7$. Then we have the following nonlinear equation:

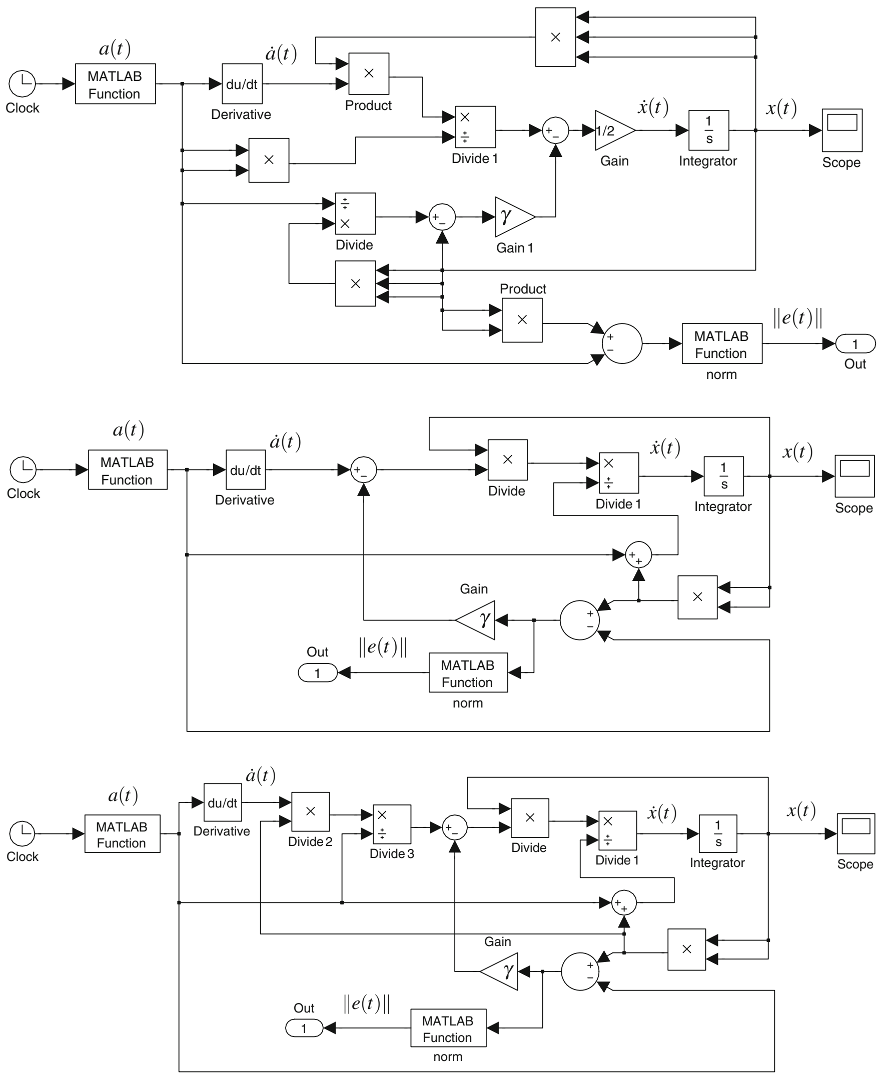
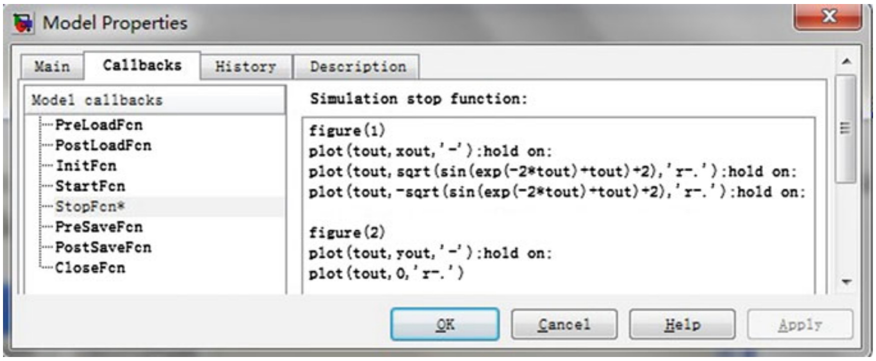


Fig. 3.6 Simulink modeling of ZD models (3.11)–(3.13) for time-varying square root finding

$$f(x(t),t) = x^2(t) - 2\cos(5t) - 3\sin(\cos(2t)) - 7 = 0. \tag{3.14}$$

The proposed ZD models (3.8)–(3.13) are exploited to solve the above time-varying square root problem (3.14), and the corresponding modeling results are illustrated in Fig. 3.8. For the convenience of observation, we randomly generate five positive and five negative initial states $x(0)$, respectively, within [3, 4] and [−3, −4] for each ZD model, which are the intervals around the theoretical initial solutions ±3.3948.



Simulation stop function:

```
figure(1)
plot(tout,xout,'-');hold on;
plot(tout,sqrt(sin(exp(-2*tout)+tout)+2),'r-.');hold on;
plot(tout,-sqrt(sin(exp(-2*tout)+tout)+2),'r-.');hold on;

figure(2)
plot(tout,yout,'-');hold on;
plot(tout,0,'r-.')
```

Fig. 3.7 Necessary "StopFcn" code of "Callbacks" in dialog box "More Properties"

As shown in Fig. 3.8, the neural states of the proposed ZD models denoted by solid curves converge to the theoretical time-varying solutions denoted by dash-dotted curves. These results have substantiated the efficacy of ZD models (3.8)–(3.13) for solving time-varying square root problem (with an appropriate value of γ). It is worth mentioning that, even if we set the initial state $x(0) \neq 0$ far away from the theoretical initial solution, the neural states of the proposed ZD models [i.e., $x(t)$] can also converge to the theoretical time-varying solutions of (3.14) rapidly.

Example 3.2 In this example, we discuss how the value of γ affects the convergence performance of the proposed ZD models.

Let us exploit ZD models (3.8) and (3.12) (as examples) to solve the following time-varying square root problem with $\gamma = 10$, 100 and 1000, respectively:

$$f(x(t),t) = x^2(t) - \sin(\exp(-2t)+t) - 2 = 0. \tag{3.15}$$

Starting from randomly-generated initial states $x(0)$ and using different values of γ, we have the convergence of modeling residual errors $\|e(t)\|$ shown in Figs. 3.9 and 3.10 by using the proposed ZD models (3.8) and (3.12), respectively. Note that symbol $\|\cdot\|$ denotes the Euclidean norm for a vector (which, in this situation, denotes the absolute value of a scalar).

As seen from Fig. 3.9, as γ increases from 10 to 100 and to 1000, the order of the maximal steady-state modeling residual error (MSSMRE) of ZD model (3.8) decreases from about 10^{-2} to 10^{-3} and to 10^{-4}, and meanwhile, the convergence time is expedited from around 0.5 to 0.05 and to 0.005 s. That is, ZD model (3.8) has an exponential-convergence property, which can be expedited effectively by increasing the value of γ. Thus, we should set the value of γ appropriately large not only to decrease the computational error, but also to shorten the convergence time. For ZD model (3.12), we have similar conclusions, which is shown from Fig. 3.10. Therefore, we have the conclusion that design-parameter γ plays an important role in ZD models (3.8) and (3.12) on the convergence performance (including accuracy),

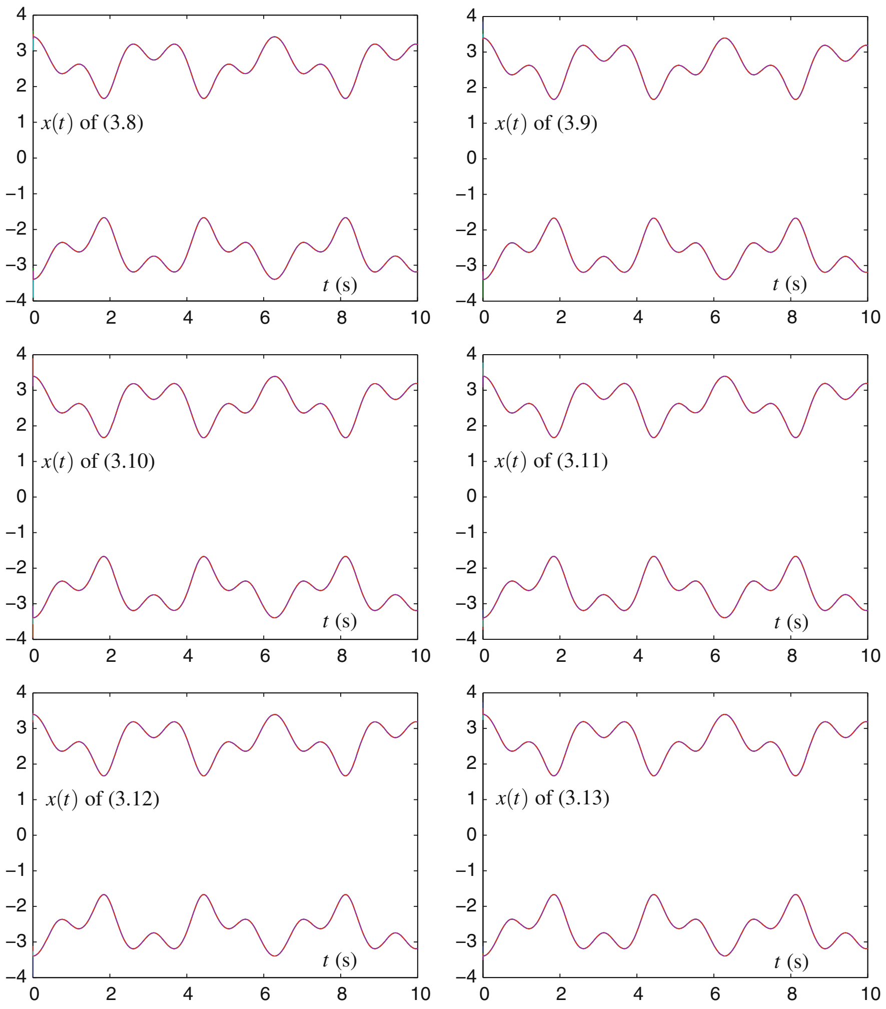


Fig. 3.8 State trajectories of ZD models (3.8)–(3.13) for solving the time-varying square root problem (3.14)

which implies that the convergence performance of the proposed ZD models can be improved by increasing the value of γ. Note that, for other ZD models [i.e., (3.9)–(3.11) and (3.13)], we have similar conclusions by observing the related modeling results, which are omitted due to results' similarity. Being a topic of exercise, the corresponding modeling verifications of ZD models (3.9)–(3.11) and (3.13) are left for interested readers.

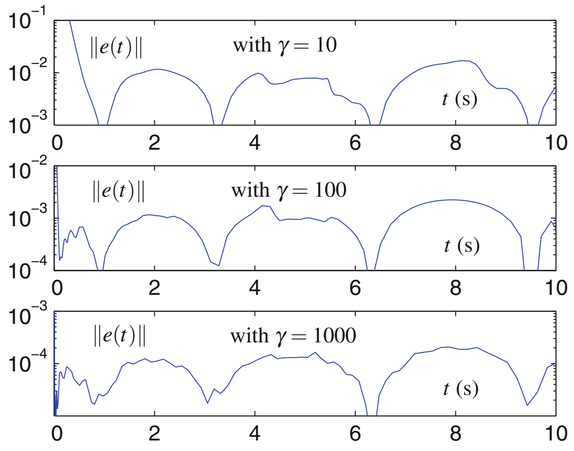
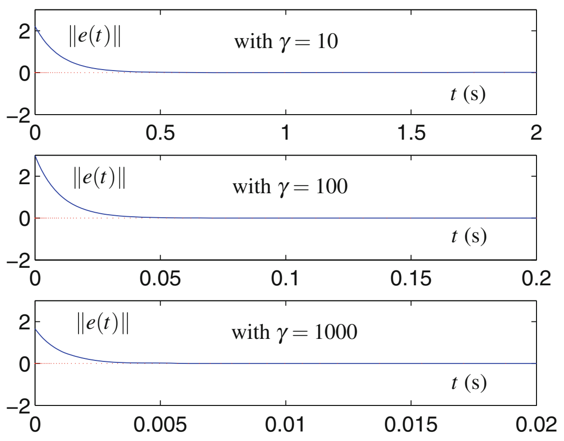


Fig. 3.9 Modeling residual errors $\|e(t)\|$ of ZD model (3.8) with different values of γ for solving the time-varying square root problem (3.15)



In summary, the above modeling results with different illustrative examples have substantiated the efficacy of the proposed ZD models (3.8)–(3.13) for time-varying square root finding.

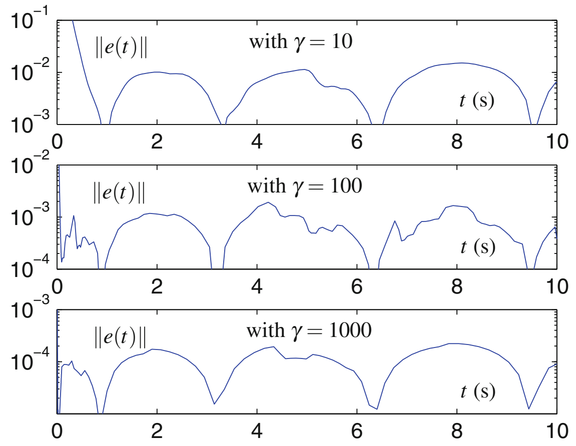
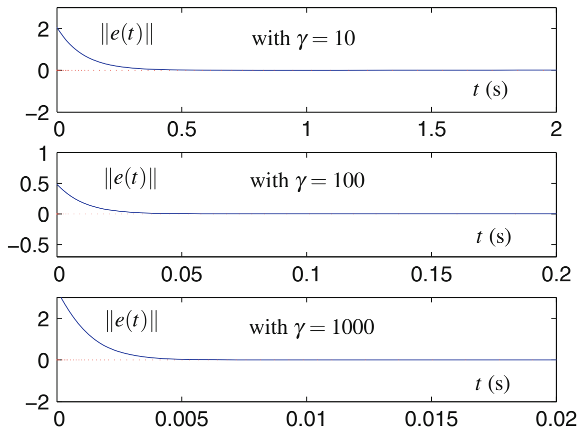


Fig. 3.10 Modeling residual errors $\|e(t)\|$ of ZD model (3.12) with different values of γ for solving the time-varying square root problem (3.15)



3.5 Summary

In this chapter, based on different ZFs (3.2)–(3.7) being the error-monitoring functions, different ZD models (3.8)–(3.13) have been proposed, generalized, developed, and investigated for time-varying square root finding. Then, for possible hardware implementation based on electronic circuits, the MATLAB Simulink modeling of the proposed ZD models has been presented. Note that this modeling technique has drastically reduced the computer-programming efforts. Through illustrative computer-modeling examples, the efficacy of the proposed ZD models has been further substantiated for time-varying square root finding.

References

1. IEEE (1985) IEEE standard for binary floating-point arithmetic. IEEE standard 754. IEEE Computer Society
2. Mathews JH, Fink KD (2005) Numerical methods using MATLAB. Publishing House of Electronics Industry, Beijing
3. Lin C (2005) Numerical computation methods. Science Press, Beijing
4. Majerski S (1985) Square-rooting algorithms for high-speed digital circuits. IEEE Trans Comput C-34(8):724–733
5. Takahashi D (2000) Implementation of multiple-precision parallel division and square root on distributed-memory parallel computers. In: Proceedings of international workshops on parallel processing, pp 229–235
6. Kong F, Cai Z, Yu J, Li DX (2006) Improved generalized Atkin algorithm for computing square roots in finite fields. Inf Process Lett 98(1):1–5
7. Pineiro JA, Bruguera JD (2002) High-speed double-precision computation of reciprocal, division, square root, and inverse square root. IEEE Trans Comput 51(12):1377–1388
8. Ercegovac MD, Lang T, Muller JM, Tisserand A (2000) Reciprocation, square root, inverse square root, and some elementary functions using small multipliers. IEEE Trans Comput 49(7):628–637
9. Zhang YN, Leithead WE, Leith DJ (2005) Time-series Gaussian process regression based on Toeplitz computation of $O(N^2)$ operations and $O(N)$-level storage. In: Proceedings of the 44th IEEE international conference on decision and control, pp 3711–3716
10. Mead C (1989) Analog VLSI and neural systems. Addison-Wesley Longman, Boston
11. Zhang Y, Ma W, Li K, Yi C (2008) Brief history and prospect of coprocessors. China Acad J Electron Publ House 13:115–117
12. Zhang Y, Ke Z, Xu P, Yi C (2010) Time-varying square roots finding via Zhang dynamics versus gradient dynamics and the former's link and new explanation to Newton-Raphson iteration. Inf Process Lett 110(24):1103–1109
13. Zhang Y, Yin Y, Guo D, Li W, Ke Z (2013) Different Zhang functions leading to different ZD models illustrated via time-varying square roots finding. In: Proceedings of the 4th international conference on intelligent control and information processing, pp 277–282
14. Zhang Y, Yi C, Guo D, Zheng J (2011) Comparison on Zhang neural dynamics and gradient-based neural dynamics for online solution of nonlinear time-varying equation. Neural Comput Appl 20(1):1–7
15. Zhang Y, Yi C (2011) Zhang neural networks and neural-dynamic method. Nova Science Publishers, New York
16. Bushard LB (1983) A minimum table size result for higher radix nonrestoring division. IEEE Trans Comput C-32(6):521–526
17. Trivedi K, Ercegovac M (1977) On-line algorithms for division and multiplication. IEEE Trans Comput C-26(7):681–687
18. Ansari MS, Rahman SA (2011) DVCC-based non-linear feedback neural circuit for solving system of linear equations. Circuits, Syst Signal Process 30(5):1029–1045
19. Shanblatt MA (2005) A Simulink-to-FPGA implementation tool for enhanced design flow. In: Proceedings of the IEEE international conference on microelectronic systems education, pp 89–90

Part II
Vector-Valued ZF in Real Domain

Chapter 4
System of Time-Varying Linear Equations

Abstract In this chapter, by following the idea of ZF, two ZD models are proposed, generalized, developed, and investigated to solve the system of time-varying linear equations. It is theoretically proved that such two ZD models globally and exponentially converge to the theoretical time-varying solution of system of time-varying linear equations. Then, we conduct extensive simulations using such two ZD models. The simulation results substantiate the theoretical analysis and the efficacy of the proposed ZD models for solving the system of time-varying linear equations.

4.1 Introduction

The problem of solving the system of linear equations (including matrix inversion problems as a closely related topic) is considered to be one of the basic problems widely encountered in science and engineering. It is usually an essential part of many solutions; e.g., as preliminary steps for optimization [1], signal processing [2], electromagnetic systems [3], and robots' inverse kinematics [4]. Due to the important role, many approaches (including numerical algorithms and neural-dynamics methods) have thus been developed, analyzed, and investigated for solving the system of linear equations [5–9]. Note that almost all of the reported methods are theoretically/intrinsically designed for solving the system of time-invariant linear equations. Therefore, these methods may be less accurate and effective enough when they are exploited directly to solve the system of time-varying linear equations [8, 9].

To lay a basis for further discussion, the following system of time-varying linear equations is presented [8, 9]:

$$A(t)\mathbf{x}(t) = \mathbf{b}(t) \in \mathbb{R}^m,\ t \in [0, +\infty), \tag{4.1}$$

where $A(t) \in \mathbb{R}^{m\times n}$ is the smoothly time-varying full-rank coefficient matrix, $\mathbf{b}(t) \in \mathbb{R}^m$ is the smoothly time-varying coefficient vector, and $\mathbf{x}(t) \in \mathbb{R}^n$ is the unknown vector that needs to be obtained. Without loss of generality, $A(t)$ and $\mathbf{b}(t)$, together with their time derivatives $\dot{A}(t)$ and $\dot{\mathbf{b}}(t)$, are assumed to be known or can be estimated

Y. Zhang and D. Guo, *Zhang Functions and Various Models*,
DOI 10.1007/978-3-662-47334-4_4

accurately. Besides, we have the following descriptions about the general behavior for the system of linear equations (4.1) [note that $A(t)$ is of full rank].

- When $m < n$, (4.1) has infinitely many solutions, and it is thus known as the underdetermined system of linear equations [10–12].
- When $m = n$, (4.1) has a single unique solution [5, 8, 9].
- When $m > n$, (4.1) has no solution, and it is thus known as the overdetermined system of linear equations [13–15].

By realizing such a classification, in this chapter, we only focus on the investigation of solving (4.1) under the situation of $m = n$ [i.e., $A(t)$ is a nonsingular square matrix]; while the investigation of solving (4.1) under the situation of $m \neq n$ (including $m < n$ and $m > n$) is conducted in the ensuing chapter (i.e., Chap. 5).

More specifically, in this chapter, focusing on solving the system of time-varying linear equations $A(t)\mathbf{x}(t) = \mathbf{b}(t)$ with $A(t) \in \mathbb{R}^{n\times n}$ and $\mathbf{b}(t) \in \mathbb{R}^n$ [i.e., (4.1) under the situation of $m = n$], we propose, generalize, develop, and investigate two different ZD models by defining two different ZFs as the error-monitoring functions. Then, theoretical results are given to show that such two ZD models globally and exponentially converge to the theoretical time-varying solution of system of time-varying linear equations. Two illustrative examples are provided and computer simulation results further substantiate the efficacy of the proposed ZD models for solving the system of time-varying linear equations.

4.2 ZFs and ZD Models

In this section, by defining two different ZFs, two ZD models are proposed for solving the system of time-varying linear equations (4.1).

In view of that (4.1) is depicted in the matrix–vector form (which is different from those scalar forms presented in the previous chapters), we denote the corresponding ZF by $\mathbf{e}(t)$ with $\dot{\mathbf{e}}(t)$ being its time derivative. Note that, in Chaps. 4–6, $\mathbf{e}(t)$ and $\dot{\mathbf{e}}(t)$ are used as the notations of the vector-valued ZF and its time derivative, respectively. Thus, the ZD design formula (1.2) presented in Chap. 1 is generalized as follows (i.e., the vector form) [9]:

$$\dot{\mathbf{e}}(t) = \frac{\mathrm{d}\mathbf{e}(t)}{\mathrm{d}t} = -\gamma\mathbf{e}(t), \tag{4.2}$$

where design parameter $\gamma \in \mathbb{R}$ is defined the same as before. Based on different ZFs and the ZD design formula (4.2), the resultant ZD models are developed for solving the system of time-varying linear equations (4.1).

4.2.1 The First ZF and ZD Model

In order to solve the system of time-varying linear equations (4.1), the first ZF (i.e., a vector-valued lower-unbounded error function) is defined as follows:

$$\mathbf{e}(t) = A(t)\mathbf{x}(t) - \mathbf{b}(t) \in \mathbb{R}^n. \tag{4.3}$$

With ZF (4.3), expanding the ZD design formula (4.2), and in view of $\dot{\mathbf{e}}(t) = A(t)\dot{\mathbf{x}}(t) + \dot{A}(t)\mathbf{x}(t) - \dot{\mathbf{b}}(t)$, we obtain

$$A(t)\dot{\mathbf{x}}(t) = -\dot{A}(t)\mathbf{x}(t) + \dot{\mathbf{b}}(t) - \gamma(A(t)\mathbf{x}(t) - \mathbf{b}(t)), \tag{4.4}$$

which is the ZD model based on ZF (4.3) for solving system of time-varying linear equations. Besides, the circuit schematic of ZD model (4.4) is depicted in Fig. 4.1, which is an important and necessary step for the final hardware implementation of the neural dynamics.

4.2.2 The Second ZF and ZD Model

Before defining the second ZF, an important theorem is presented to lay a basis for further discussion as follows:

Theorem 4.1 *The time derivative of the time-varying matrix inverse $A^{-1}(t)$ is formulated as $\dot{A}^{-1}(t) = \mathrm{d}A^{-1}(t)/\mathrm{d}t = -A^{-1}(t)\dot{A}(t)A^{-1}(t)$.*

Proof It follows from $A(t)A^{-1}(t) = I \in \mathbb{R}^{n\times n}$ (with I being the identity matrix) that

$$\frac{\mathrm{d}\left(A(t)A^{-1}(t)\right)}{\mathrm{d}t} = \frac{\mathrm{d}I}{\mathrm{d}t} = \mathbf{0} \in \mathbb{R}^{n\times n}.$$

Expanding the above equation, we obtain

$$\frac{\mathrm{d}A(t)}{\mathrm{d}t}A^{-1}(t) + A(t)\frac{\mathrm{d}A^{-1}(t)}{\mathrm{d}t} = \mathbf{0} \in \mathbb{R}^{n\times n},$$

which is further rewritten as

$$A(t)\frac{\mathrm{d}A^{-1}(t)}{\mathrm{d}t} = -\frac{\mathrm{d}A(t)}{\mathrm{d}t}A^{-1}(t) = -\dot{A}(t)A^{-1}(t).$$

Then, we have

$$\frac{\mathrm{d}A^{-1}(t)}{\mathrm{d}t} = -A^{-1}(t)\dot{A}(t)A^{-1}(t),$$

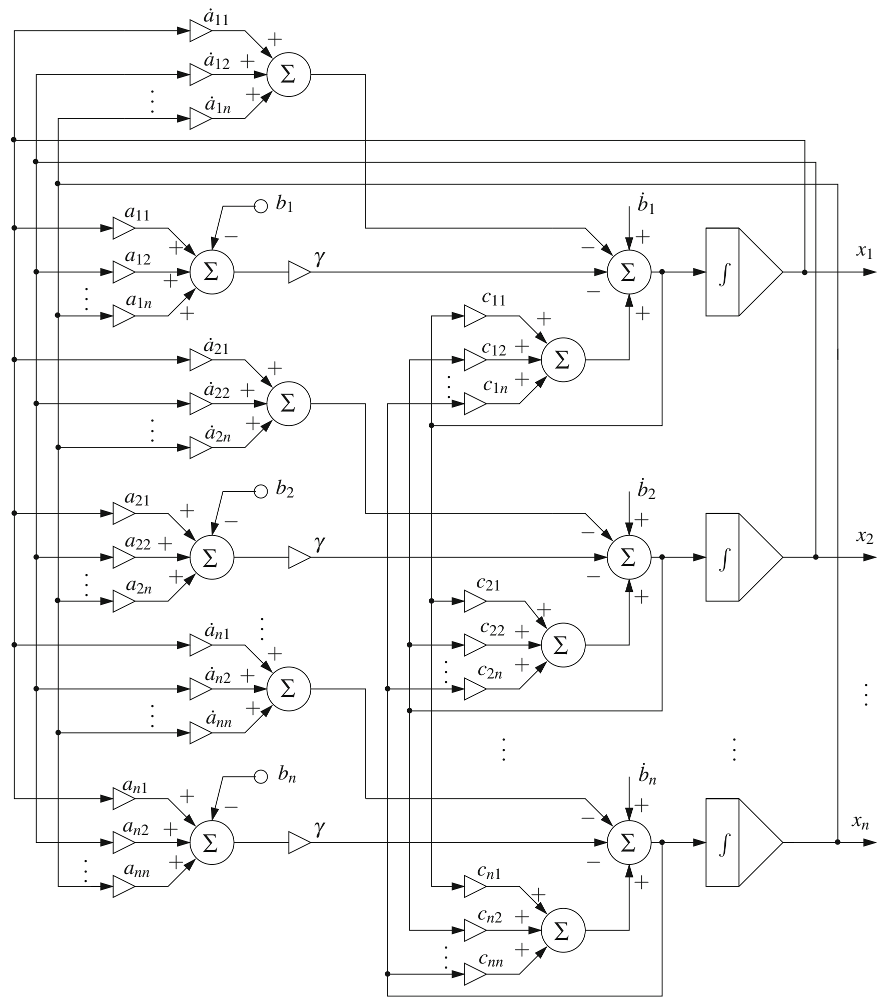


Fig. 4.1 The circuit schematic which realizes the ZD model (4.4), where such a model has been rewritten as $\dot{\mathbf{x}}(t) = C(t)\dot{\mathbf{x}}(t) - \dot{A}(t)\mathbf{x}(t) + \dot{\mathbf{b}}(t) - \gamma(A(t)\mathbf{x}(t) - \mathbf{b}(t))$ with $C(t) = I - A(t) \in \mathbb{R}^{n\times n}$

i.e.,

$$\dot{A}^{-1}(t) = -A^{-1}(t)\dot{A}(t)A^{-1}(t).$$

The proof is thus complete. □

Having the above theoretical result, we can define the second ZF as below:

$$\mathbf{e}(t) = \mathbf{x}(t) - A^{-1}(t)\mathbf{b}(t). \tag{4.5}$$

Then, in view of the above definition (4.5) and $\dot{A}^{-1}(t) = -A^{-1}(t)\dot{A}(t)A^{-1}(t)$, we have the following ZD model by expanding ZD design formula (4.2):

$$\dot{\mathbf{x}}(t) - \left(-A^{-1}(t)\dot{A}(t)A^{-1}(t)\mathbf{b}(t) + A^{-1}(t)\dot{\mathbf{b}}(t)\right) = -\gamma\left(\mathbf{x}(t) - A^{-1}(t)\mathbf{b}(t)\right),$$

and equivalently

$$A(t)\dot{\mathbf{x}}(t) = -\dot{A}(t)A^{-1}(t)\mathbf{b}(t) + \dot{\mathbf{b}}(t) - \gamma\left(A(t)\mathbf{x}(t) - \mathbf{b}(t)\right). \tag{4.6}$$

Thus, based on the second ZF (4.5), the second ZD model (4.6) is obtained for solving the system of time-varying linear equations (4.1).

4.3 Theoretical Results and Analyses

In this section, theoretical results and analyses are presented, which show the convergence performance of the proposed two ZD models (4.4) and (4.6) on solving the system of time-varying linear equations (4.1).

Theorem 4.2 *Given a smoothly time-varying nonsingular coefficient matrix $A(t) \in \mathbb{R}^{n\times n}$ and a smoothly time-varying vector $\mathbf{b}(t) \in \mathbb{R}^n$ in (4.1), the state vector $\mathbf{x}(t) \in \mathbb{R}^n$ of ZD model (4.4), starting from a randomly-generated initial state $\mathbf{x}(0)$, converges globally and exponentially to the theoretical time-varying solution $\mathbf{x}^*(t)$ of (4.1) with time.*

Proof Let $\tilde{\mathbf{x}}(t) = \mathbf{x}(t) - \mathbf{x}^*(t)$ denote the difference between the time-varying solution $\mathbf{x}(t)$ generated by the proposed ZD model (4.4) and the theoretical time-varying solution $\mathbf{x}^*(t)$ of (4.1). Then, we have

$$\mathbf{x}(t) = \tilde{\mathbf{x}}(t) + \mathbf{x}^*(t).$$

Substituting the above equation to (4.4); and in view of the equation $A(t)\mathbf{x}^*(t) - \mathbf{b}(t) = \mathbf{0} \in \mathbb{R}^n$ and its time derivative $A(t)\dot{\mathbf{x}}^*(t) + \dot{A}(t)\mathbf{x}^*(t) - \dot{\mathbf{b}}(t) = \mathbf{0}$, we further know that $\tilde{\mathbf{x}}(t)$ is the solution to the following dynamic equation:

$$A(t)\dot{\tilde{\mathbf{x}}}(t) = -\dot{A}(t)\tilde{\mathbf{x}}(t) - \gamma A(t)\tilde{\mathbf{x}}(t), \tag{4.7}$$

where $\dot{\tilde{\mathbf{x}}}(t)$ is the time derivative of $\tilde{\mathbf{x}}(t)$.

Since $\mathbf{e}(t) = A(t)\mathbf{x}(t) - \mathbf{b}(t) = A(t)(\tilde{\mathbf{x}}(t) + \mathbf{x}^*(t)) - \mathbf{b}(t) = A(t)\tilde{\mathbf{x}}(t)$, (4.7) is rewritten equivalently as $\dot{\mathbf{e}}(t) = -\gamma\mathbf{e}(t)$, which is a compact vector form of the following set of n equations (where $i = 1, 2, \ldots, n$):

$$\dot{e}_i = -\gamma e_i \in \mathbb{R}. \tag{4.8}$$

Then, we define a Lyapunov function candidate $v = e_i^2/2 \geqslant 0$ for the ith subsystem (4.8). Evidently, such a Lyapunov function is positive-definite, because $v > 0$ for any $e_i \neq 0$, and $v = 0$ only for $e_i = 0$. Next, its time derivative is obtained as below:

$$\dot{v} = \frac{\mathrm{d}v}{\mathrm{d}t} = e_i \dot{e}_i = -\gamma e_i^2 \leqslant 0,$$

which guarantees the final negative-definiteness of $\dot{v}$. That is to say, $\dot{v} < 0$ for any $e_i \neq 0$, and $\dot{v} = 0$ only for $e_i = 0$. In addition, as $e_i \to \infty$, $v \to \infty$. According to Lyapunov theory, e_i globally converges to zero for any $i \in \{1, 2, \ldots, n\}$. Therefore, in view of $\mathbf{e}(t) = A(t)\tilde{\mathbf{x}}(t)$ and the invertible time-varying matrix $A(t)$, we have $\tilde{\mathbf{x}}(t) \to 0$ as $t \to \infty$. In other words, starting from a randomly-generated initial state $\mathbf{x}(0)$, the state vector $\mathbf{x}(t)$ of ZD model (4.4) converges globally to the theoretical time-varying solution $\mathbf{x}^*(t) \in \mathbb{R}^n$.

Next, we prove the exponential convergence performance of ZD model (4.4). In view of (4.8), we obtain its analytic solution in the compact vector form as follows:

$$\mathbf{e}(t) = \mathbf{e}(0)\exp(-\gamma t).$$

Then, we have

$$\|\mathbf{e}(t)\|_2 = \|\mathbf{e}(0)\|_2 \exp(-\gamma t),$$

where $\|\cdot\|_2$ denotes the two norm of a vector. Evidently, as $t \to \infty$, $\|\mathbf{e}(t)\|_2 = \|A(t)\tilde{\mathbf{x}}(t)\|_2 \to 0$ exponentially with rate γ. Given $\alpha > 0$ as the minimum eigenvalue of $A^{\mathrm{T}}(t)A(t)$, we have $\|A(t)\tilde{\mathbf{x}}(t)\|_2 = \tilde{\mathbf{x}}^{\mathrm{T}}(t)A^{\mathrm{T}}(t)A(t)\tilde{\mathbf{x}}(t) \geqslant \alpha\tilde{\mathbf{x}}^{\mathrm{T}}(t)\tilde{\mathbf{x}}(t) = \alpha\|\tilde{\mathbf{x}}(t)\|_2$, where superscript $^{\mathrm{T}}$ denotes the transpose operator. This implies that, starting from any randomly-generated initial state $\mathbf{x}(0)$, the state vector $\mathbf{x}(t)$ of ZD model (4.4) converges exponentially to the theoretical time-varying solution $\mathbf{x}^*(t)$ of (4.1) as time t goes on. Based on the above analysis, the proof is thus complete. □

Theorem 4.3 *Given a smoothly time-varying nonsingular coefficient matrix $A(t) \in \mathbb{R}^{n\times n}$ and a smoothly time-varying vector $\mathbf{b}(t) \in \mathbb{R}^n$ in (4.1), the state vector $\mathbf{x}(t) \in \mathbb{R}^n$ of ZD model (4.6), starting from a randomly-generated initial state $\mathbf{x}(0)$, converges globally and exponentially to the theoretical time-varying solution $\mathbf{x}^*(t)$ of (4.1) with rate $\gamma > 0$, as time t goes on.*

Proof Let $\tilde{\mathbf{x}}(t) = \mathbf{x}(t) - \mathbf{x}^*(t)$. Substituting it to (4.6); and considering equation $\mathbf{x}^*(t) - A^{-1}(t)\mathbf{b}(t) = \mathbf{0}$ and its time derivative $\dot{\mathbf{x}}^*(t) + A^{-1}(t)\dot{A}(t)A^{-1}(t)\mathbf{b}(t) - A^{-1}(t)\,\dot{\mathbf{b}}(t) = \mathbf{0}$, we further know that $\tilde{\mathbf{x}}(t)$ is the solution to the ensuing dynamics

$$\dot{\tilde{\mathbf{x}}}(t) = -\gamma\tilde{\mathbf{x}}(t) \in \mathbb{R}^n. \tag{4.9}$$

Thus, we can define a Lyapunov function candidate $v = \tilde{x}_i^2/2 \geqslant 0$ for the ith subsystem of (4.9), where $\tilde{x}_i$ denotes the ith element of $\tilde{\mathbf{x}}(t)$ (with $i = 1, 2, \ldots, n$).

Evidently, such a Lyapunov function is positive-definite, because $v > 0$ for any $\tilde{x}_i \neq 0$, and $v = 0$ only for $\tilde{x}_i = 0$. Then, its time derivative is obtained as below:

$$\dot{v} = \frac{\mathrm{d}v}{\mathrm{d}t} = \tilde{x}_i \dot{\tilde{x}}_i = -\gamma \tilde{x}_i^2 \leqslant 0,$$

which guarantees the final negative-definiteness of $\dot{v}$. That is to say, $\dot{v} < 0$ for any $\tilde{x}_i \neq 0$, and $\dot{v} = 0$ only for $\tilde{x}_i = 0$. In addition, as $\tilde{x}_i \to \infty$, $v \to \infty$. By Lyapunov theory, $\tilde{x}_i$ globally converges to zero for any $i \in \{1, 2, \ldots, n\}$. Therefore, we have $\tilde{\mathbf{x}}(t) \to \mathbf{0}$ as $t \to \infty$. That is to say, starting from a randomly-generated initial state $\mathbf{x}(0)$, state vector $\mathbf{x}(t)$ of ZD model (4.6) converges globally to the theoretical time-varying solution. Now, we prove the exponential convergence of ZD model (4.6).

In view of (4.9), we can obtain its analytic solution as follows:

$$\tilde{\mathbf{x}}(t) = \tilde{\mathbf{x}}(0)\exp(-\gamma t),$$

and

$$\|\tilde{\mathbf{x}}(t)\|_2 = \|\tilde{\mathbf{x}}(0)\|_2 \exp(-\gamma t),$$

which means that, as $t \to \infty$, $\|\tilde{\mathbf{x}}(t)\|_2 = \|\mathbf{x}(t) - \mathbf{x}^*(t)\|_2 \to 0$ exponentially with rate γ. That is to say, starting from a randomly-generated initial state $\mathbf{x}(0)$, state vector $\mathbf{x}(t)$ of ZD model (4.6) converges globally and exponentially to theoretical time-varying solution $\mathbf{x}^*(t)$ of (4.1) with rate $\gamma > 0$. The proof is thus complete. □

According to the theoretical analysis above, we can draw the conclusion that the proposed ZD models globally and exponentially converge to the theoretical solution of system of time-varying linear equations. Note that the convergence time of both ZD models can be expedited as the value of design parameter γ increases. Specifically, the convergence time decreases in an exponential manner with the γ value increasing. That is to say, the convergence time of ZD model (4.4) and ZD model (4.6) for solving the system of time-varying linear equations (4.1) is decreasing as the value of γ increases. Therefore, design parameter γ plays an important role in ZD models and thus should be selected appropriately large to satisfy the convergence rate in practice.

4.4 Illustrative Examples

In the previous sections, two different ZFs have been presented, which generate two ZD models for solving the system of time-varying linear equations (4.1). In addition to detailed design process of ZD models, their excellent convergence performances are analyzed in details. In this section, two illustrative examples are provided for substantiating the efficacy of such two ZD models.

Example 4.1 Now let us consider the following time-varying coefficients of (4.1) with $A(t)$ and $\mathbf{b}(t)$ being, respectively,

$$A(t) = \begin{bmatrix} \sin(3t) & \cos(3t) \\ -\cos(3t) & \sin(3t) \end{bmatrix} \in \mathbb{R}^{2\times 2} \text{ and } \mathbf{b}(t) = \begin{bmatrix} \sin(3t)+1 \\ -\cos(3t) \end{bmatrix} \in \mathbb{R}^2.$$

Then, the theoretical time-varying solution $\mathbf{x}^*(t)$ of (4.1) in this situation is obtained as below:

$$\mathbf{x}^*(t) = \begin{bmatrix} \sin(3t) & -\cos(3t) \\ \cos(3t) & \sin(3t) \end{bmatrix} \begin{bmatrix} \sin(3t)+1 \\ -\cos(3t) \end{bmatrix}.$$

Since we have got the theoretical time-varying solution of (4.1), we can use it as a criterion to substantiate the effectiveness of the proposed ZD models (4.4) and (4.6). The corresponding simulation results are shown in Figs. 4.2 and 4.3.

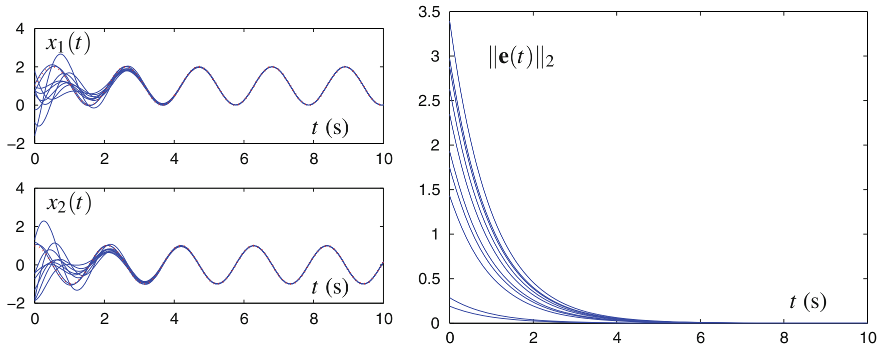


Fig. 4.2 Transient behaviors of neural state $\mathbf{x}(t)$ and residual error $\|\mathbf{e}(t)\|_2 = \|A(t)\mathbf{x}(t) - \mathbf{b}(t)\|_2$ synthesized by ZD model (4.4) starting with ten randomly-generated initial states in Example 4.1

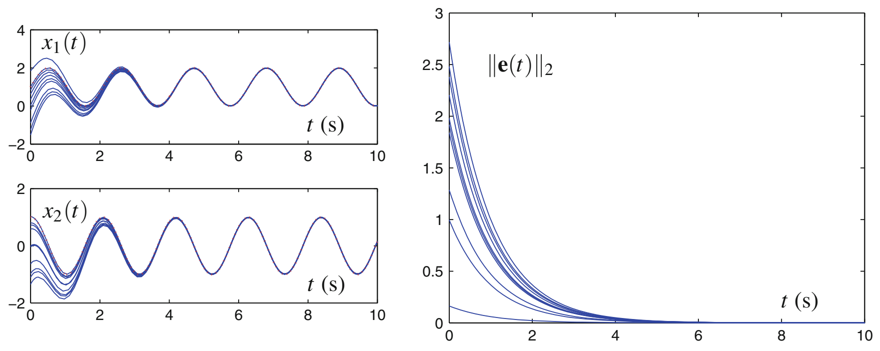


Fig. 4.3 Transient behaviors of neural state $\mathbf{x}(t)$ and residual error $\|\mathbf{e}(t)\|_2 = \|A(t)\mathbf{x}(t) - \mathbf{b}(t)\|_2$ synthesized by ZD model (4.6) starting with ten randomly-generated initial states in Example 4.1

First, we investigate the convergence performance of ZD model (4.4) with $\gamma = 1$. Starting from ten randomly-generated initial states, the transient behavior of neural state $\mathbf{x}(t) \in \mathbb{R}^2$ is shown in the left graph of Fig. 4.2. As seen from it, neural states of ZD model (4.4) all converge to the theoretical time-varying solution of (4.1) accurately. That is, $\mathbf{x}(t)$ shown in the left graph of Fig. 4.2 is an exact solution of system of time-varying linear equations (4.1). In addition, the right graph of Fig. 4.2 shows the transient behavior of residual error $\|\mathbf{e}(t)\|_2 = \|A(t)\mathbf{x}(t) - \mathbf{b}(t)\|_2$ corresponding to $\mathbf{x}(t)$ synthesized by ZD model (4.4). It follows from the right graph of Fig. 4.2 that the residual error synthesized by (4.4) decreases exponentially to zero as time t goes on. These results coincide with the theoretical analysis of Theorem 4.2. Thus, the efficacy of the proposed ZD model (4.4) for solving the system of time-varying linear equations (4.1) is substantiated primarily.

Then, we investigate the convergence performance of ZD model (4.6) under the same conditions as before. For demonstrating the effectiveness of ZD model (4.6), the left graph of Fig. 4.3 shows the transient behavior of state vector $\mathbf{x}(t) \in \mathbb{R}^2$. As seen from it, neural states of (4.6) all converge to the theoretical time-varying solution rapidly. The right graph of Fig. 4.3 further shows the corresponding transient behavior of residual error $\|\mathbf{e}(t)\|_2 = \|A(t)\mathbf{x}(t) - \mathbf{b}(t)\|_2$. Evidently, from the right graph of Fig. 4.3, we know that the residual error synthesized by ZD model (4.6) converges exponentially to zero. These results substantiate the efficacy of the proposed ZD model (4.6) for solving system of time-varying linear equations (4.1) (as well as coincide with the theoretical analysis of Theorem 4.3).

Example 4.2 For further demonstrating the effectiveness of the proposed ZD models (4.4) and (4.6), let us consider the following time-varying coefficients of (4.1):

$$A(t) = \begin{bmatrix} 3+\sin(5t) & \cos(5t) & 0.5\cos(5t) \\ \cos(5t) & 3+\sin(5t) & \cos(5t) \\ 0.5\cos(5t) & \cos(5t) & 3+\sin(5t) \end{bmatrix} \in \mathbb{R}^{3\times 3} \text{ and}$$
$$\mathbf{b}(t) = \begin{bmatrix} \cos(5t) \\ \sin(5t) \\ -\sin(5t) \end{bmatrix} \in \mathbb{R}^3.$$

Similarly, we apply such two ZD models (4.4) and (4.6) with $\gamma = 1$ to solve the above system of time-varying linear equations, and the corresponding simulation results are shown in Fig. 4.4. From Fig. 4.4 which shows the transient behavior of $\|\mathbf{e}(t)\|_2 = \|A(t)\mathbf{x}(t) - \mathbf{b}(t)\|_2$, we can see that the residual errors synthesized by ZD models (4.4) and (4.6) both converge exponentially to zero as time t goes on. This also implies that the corresponding state vector $\mathbf{x}(t) \in \mathbb{R}^3$ [of (4.4) or (4.6)], starting from ten randomly-generated initial states, is an exact solution of the aforementioned system of time-varying linear equations. Thus, based on these results, the efficacy of the proposed ZD models (4.4) and (4.6) is further substantiated.

Example 4.3 As mentioned before, the convergence time of both ZD models (4.4) and (4.6) can be expedited as the value of design parameter γ increases. Thus, in

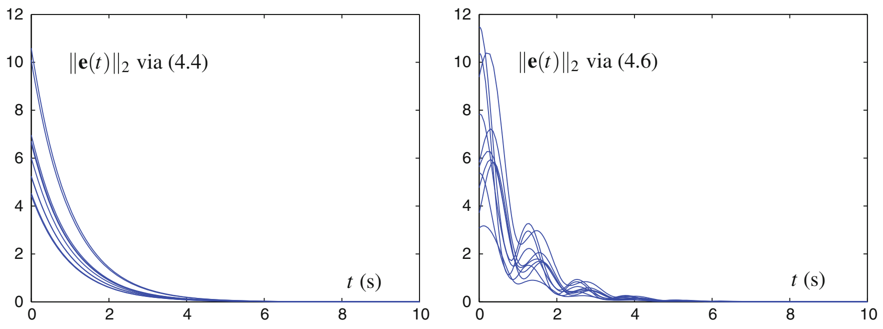


Fig. 4.4 Transient behaviors of residual errors $\|\mathbf{e}(t)\|_2 = \|A(t)\mathbf{x}(t) - \mathbf{b}(t)\|_2$ synthesized by ZD models (4.4) and (4.6) starting with the same randomly-generated initial state in Example 4.2

this example, we further investigate the convergence performance of such two ZD models using different γ values.

Let us exploit ZD models (4.4) and (4.6) to solve the system of time-varying linear equations (4.1) involved in Example 4.1 with $\gamma = 1$, 5, 10 and 20, respectively. The corresponding simulation results are shown in Fig. 4.5. As seen from Fig. 4.5, the residual errors synthesized by ZD models (4.4) and (4.6) all converge to zero, showing again the efficacy of such two ZD models. More importantly, from Fig. 4.5, we confirmedly observe that the residual errors with larger γ value converge faster than those with smaller γ value. In other words, the convergence times of ZD models (4.4) and (4.6) for solving the system of time-varying linear equations (4.1) become shorter and shorter as the value of γ increases. Therefore, we have the conclusion that design parameter γ plays an important role in ZD models (4.4) and (4.6) on the convergence performance.

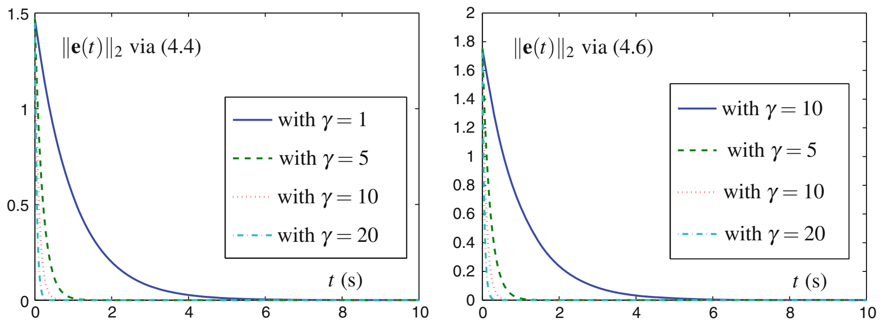


Fig. 4.5 Transient behaviors of residual errors $\|\mathbf{e}(t)\|_2 = \|A(t)\mathbf{x}(t) - \mathbf{b}(t)\|_2$ synthesized by ZD models (4.4) and (4.6) using different γ values in Example 4.3

In summary, the above simulation results with different illustrative examples have substantiated the efficacy of the proposed ZD models (4.4) and (4.6) for solving the system of time-varying linear equations (4.1).

4.5 Summary

In this chapter, by introducing two ZFs (4.3) and (4.5), two corresponding ZD models (4.4) and (4.6) have been proposed, generalized, developed, and investigated to solve system of time-varying linear equations (4.1). In addition, it has been proved that such two ZD models globally and exponentially converge to the theoretical time-varying solution of (4.1). Computer simulation results have further substantiated the theoretical analysis and the efficacy of the proposed ZD models for solving the system of time-varying linear equations.

References

1. Zhang Y (2006) Towards piecewise-linear primal neural networks for optimization and redundant robotics. In: Proceedings of IEEE international conference on networking, sensing and control, pp 374–379
2. Steriti RJ, Fiddy MA (1993) Regularized image reconstruction using SVD and a neural network method for matrix inversion. IEEE Trans Signal Process 41(10):3074–3077
3. Sarkar T, Siarkiewicz K, Stratton R (1981) Survey of numerical methods for solution of large systems of linear equations for electromagnetic field problems. IEEE Trans Antennas Propag 29(6):847–856
4. Sturges RH Jr (1988) Analog matrix inversion (robot kinematics). IEEE J Robot Autom 4(2):157–162
5. Mathews JH, Fink KD (2004) Numerical methods using MATLAB. Prentice Hall, New Jersey
6. Wang J (1992) Electronic realisation of recurrent neural work for solving simultaneous linear equations. Electron Lett 28(5):493–495
7. Zhang Y, Chen K (2008) Global exponential convergence and stability of Wang neural network for solving online linear equations. Electron Lett 44(2):145–146
8. Zhang Y, Chen K, Ma W (2007) MATLAB simulation and comparison of Zhang neural network and gradient neural network for online solution of linear time-varying equations. In: Proceedings of international conference on life system modeling and simulation, pp 450–454
9. Zhang Y, Yi C (2011) Zhang neural networks and neural-dynamic method. Nova Science Publishers, New York
10. Zhang Y, Wang Y, Jin L, Mu B, Zheng H (2013) Different ZFs leading to various ZNN models illustrated via online solution of time-varying underdetermined systems of linear equations with robotic application. Lect Notes Comput Sci 7952:481–488
11. Siciliano B, Sciavicco L, Villani L, Oriolo G (2009) Robotics: modelling, planning and control. Springer, London
12. Zhang Y, Zhang Z (2013) Repetitive motion planning and control of redundant robot manipulators. Springer, New York
13. Zhang Y, Li W, Guo D, Mu B, Zheng H (2013) Different Zhang functions leading to various ZNN models illustrated via solving the time-varying overdetermined system of linear equations. In: Proceedings of international conference on information science and technology, pp 771–776

14. Brannigan M (1982) The strict Chebyshev solution of overdetermined systems of linear equations with rank deficient matrix. Numer Math 40(3):307–318
15. Rathinavelan T, Im W (2007) Explicit treatment of force contribution from alignment tensor using overdetermined linear equations and its application in NMR structure determination. J Comput Chem 28(11):1858–1864

Chapter 5
Over-Determined and Under-Determined Systems of Time-Varying Linear Equations

Abstract In this chapter, focusing on solving over-determined system of time-varying linear equations, we first propose, generalize, develop, and investigate two ZD models based on two different ZFs. Then, by introducing another two different ZFs, another two ZD models are proposed, generalized, developed, and investigated to solve under-determined system of time-varying linear equations. Computer simulation results with different illustrative examples are presented to further substantiate the efficacy of the proposed ZD models for solving over-determined and under-determined systems of time-varying linear equations.

5.1 Introduction

Solving over-determined and under-determined systems of linear equations is widely encountered in a variety of scientific and engineering research fields [1–7]. As presented in Chap. 4, it has no solution for over-determined system of linear equations; while it has infinitely many solutions for under-determined system of linear equations. These characteristics make it difficult for solving over-determined and under-determined systems of linear equations.

Focusing on solving over-determined and under-determined systems of linear equations, many approaches (including numerical algorithms and neural-dynamics methods) have thus been developed, analyzed, and investigated [8–12]. Note that the problems of over-determined and under-determined systems of linear equations in most of these researches or investigations are static (or termed, time-invariant). This also means that almost all of these methods are theoretically/intrinsically designed for solving over-determined and under-determined systems of time-invariant linear equations. When these methods are exploited directly to solve the (over-determined or under-determined) system of time-varying linear equations, they may be less accurate and effective enough [10, 11].

In this chapter, by following the idea of ZFs, different ZD models are proposed, generalized, developed, and investigated to solve over-determined and

Y. Zhang and D. Guo, *Zhang Functions and Various Models*,
DOI 10.1007/978-3-662-47334-4_5

under-determined systems of time-varying linear equations. Specifically, we first construct two different ZD models based on two ZFs for solving over-determined system of time-varying linear equations. Then, another two different ZD models are constructed for solving under-determined system of time-varying linear equations. Four illustrative examples are provided and computer simulation results further substantiate the efficacy of the proposed ZD models for solving over-determined and under-determined systems of time-varying linear equations.

5.2 ZFs and ZD Models

In this section, by defining different ZFs, different ZD models are proposed for solving the following system of time-varying linear equations:

$$A(t)\mathbf{x}(t) = \mathbf{b}(t) \in \mathbb{R}^m,\ t \in [0, +\infty), \tag{5.1}$$

where $A(t) \in \mathbb{R}^{m\times n}$ with $m \neq n$ is the smoothly time-varying full-rank coefficient matrix, $\mathbf{b}(t) \in \mathbb{R}^m$ is the smoothly time-varying coefficient vector, and $\mathbf{x}(t) \in \mathbb{R}^n$ is the unknown vector that needs to be obtained in an error-free and real-time manner (or termed, the manner of real-time time-varying problem-solving). Note that (5.1) can be viewed as a general time-varying system of m real-valued time-varying linear equations and n real-valued time-varying variables.

To lay a basis for further discussion, the following corollary is presented, with the related proof being generalized from the proof of Theorem 4.1 and being left to interested readers to complete as a topic of exercise.

Corollary 5.1 *Consider a smoothly time-varying full-rank matrix $A(t) \in \mathbb{R}^{m\times n}$ with $m \neq n$. Let $A^+(t) \in \mathbb{R}^{n\times m}$ denote the time-varying Moore–Penrose pseudoinverse of $A(t)$. Then, the time derivative of $A^+(t)$ is formulated as $\dot{A}^+(t) = \mathrm{d}A^+(t)/\mathrm{d}t = -A^+(t)\dot{A}(t)A^+(t)$.*

5.2.1 *With $m > n$ (Over-Determined System)*

In this subsection, two different ZD models based on two ZFs are developed and investigated for solving over-determined system of time-varying linear equations, i.e., (5.1) with $m > n$. Note that, as mentioned in [6], if there exists at least one choice for the time-varying vector $\mathbf{x}(t)$ which satisfies (5.1) with $m > n$, then the over-determined system of time-varying linear equations is consistent; and if no such time-varying vector exists, then the over-determined system of time-varying linear equations is inconsistent. In this chapter, we only consider the situation of the inconsistent over-determined system of time-varying linear equations. Besides, in the study of the inconsistent over-determined system of time-varying linear equations, the two-norm technique is adopted to zero out the time-varying residual error $A(t)\mathbf{x}(t) - \mathbf{b}(t)$ as possible as we can in this chapter.

The First ZF and ZD Model In order to solve the system of time-varying linear equations (5.1) with $m > n$, the first ZF (i.e., a vector-valued lower-unbounded error function) is defined as follows:

$$\mathbf{e}(t) = A(t)\mathbf{x}(t) - \mathbf{b}(t) \in \mathbb{R}^m. \tag{5.2}$$

With ZF (5.2), by expanding the ZD design formula (4.2), we obtain the following ZD model for solving over-determined system of time-varying linear equations:

$$A^{\mathrm{T}}(t)A(t)\dot{\mathbf{x}}(t) = -A^{\mathrm{T}}(t)\dot{A}(t)\mathbf{x}(t) + A^{\mathrm{T}}(t)\dot{\mathbf{b}}(t) - \gamma A^{\mathrm{T}}(t)(A(t)\mathbf{x}(t) - \mathbf{b}(t)), \tag{5.3}$$

where $\mathbf{x}(t)$, starting from an initial condition $\mathbf{x}(0)$, is the neural state corresponding to an approximate time-varying solution (e.g., a pseudoinverse-type solution) $\mathbf{x}^*(t)$ of (5.1) with $m > n$.

The Second ZF and ZD Model To solve over-determined system of time-varying linear equations, i.e., (5.1) with $m > n$, the second ZF is defined as follows:

$$\mathbf{e}(t) = \mathbf{x}(t) - A^{+}(t)\mathbf{b}(t) \in \mathbb{R}^n. \tag{5.4}$$

where $A^{+}(t) = (A^{\mathrm{T}}(t)A(t))^{-1}A^{\mathrm{T}}(t)$ denotes the left Moore–Penrose inverse of $A(t)$.

Then, in view of (5.4) and $\dot{A}^{+}(t) = \mathrm{d}A^{+}(t)/\mathrm{d}t = -A^{+}(t)\dot{A}(t)A^{+}(t)$, we have the following ZD model by expanding ZD design formula (4.2):

$$A^{\mathrm{T}}(t)A(t)\dot{\mathbf{x}}(t) = -A^{\mathrm{T}}(t)\dot{A}(t)A^{+}(t)\mathbf{b}(t) + A^{\mathrm{T}}(t)\dot{\mathbf{b}}(t) - \gamma A^{\mathrm{T}}(t)\left(A(t)\mathbf{x}(t) - \mathbf{b}(t)\right). \tag{5.5}$$

Thus, based on the second ZF (5.4), the second ZD model (5.5) is obtained for solving over-determined system of time-varying linear equations.

Before closing this subsection of constructing ZD models (5.3) and (5.5), the block diagrams and overall Simulink models corresponding to such two ZD models are shown in Figs. 5.1, 5.2, 5.3 and 5.4, which may be useful for their future implementations on circuit systems. Besides, it is worth pointing out that the over-determined system of time-varying linear equations discussed in this chapter is inconsistent and there does not exist accurate theoretical solution for it. Thus, in the ensuing simulations, the two-norm measure is adopted to show the residual error about the obtained approximate solution of (5.1) with $m > n$, i.e., $\|A(t)\mathbf{x}(t) - \mathbf{b}(t)\|_2$.

5.2.2 With $m < n$ (Under-Determined System)

In this subsection, another two different ZD models based on two ZFs are developed and investigated for solving under-determined system of time-varying linear

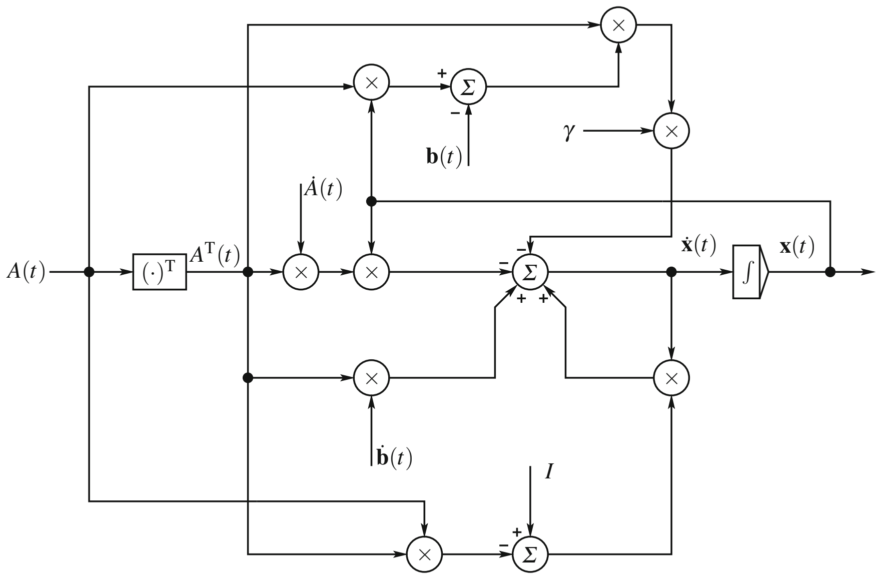


Fig. 5.1 Block diagram of ZD model (5.3) for solving over-determined system of time-varying linear equations, where I is the identity matrix

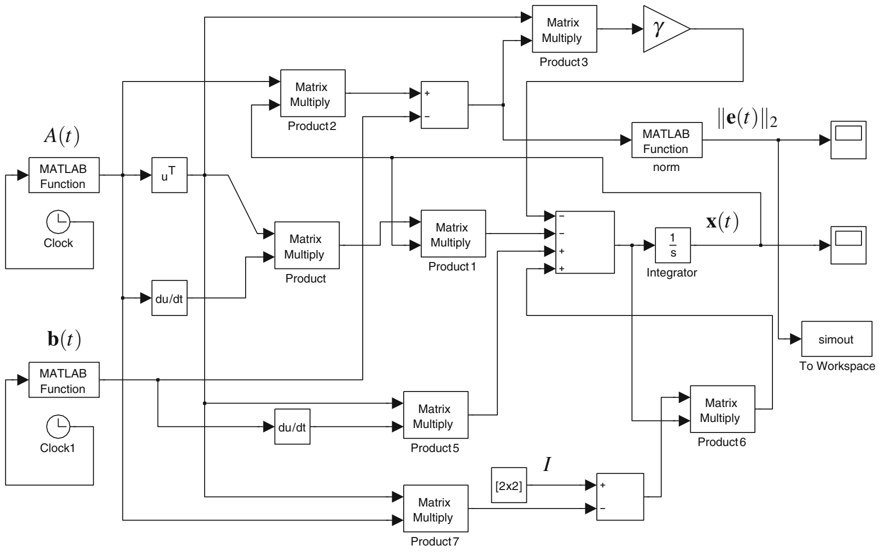


Fig. 5.2 Overall Simulink model of ZD (5.3) for solving over-determined system of time-varying linear equations

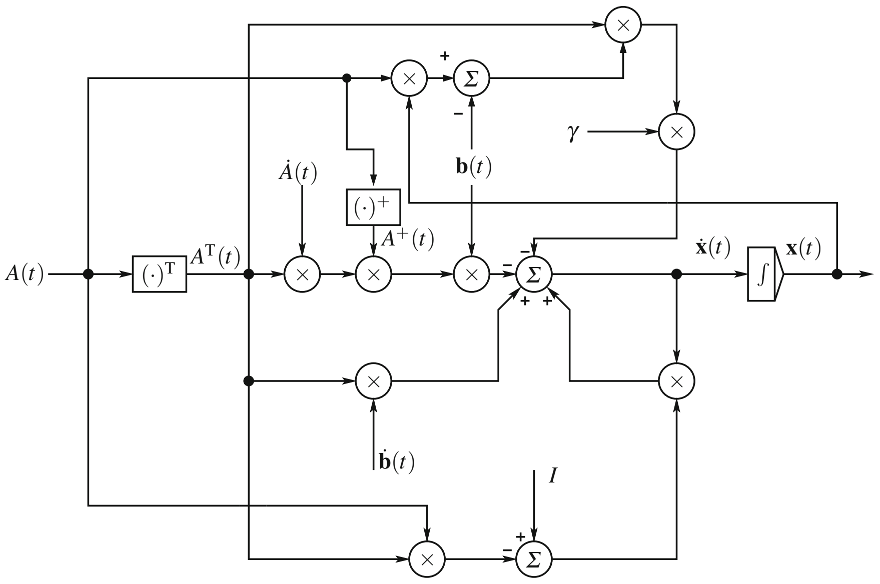


Fig. 5.3 Block diagram of ZD model (5.5) for solving over-determined system of time-varying linear equations

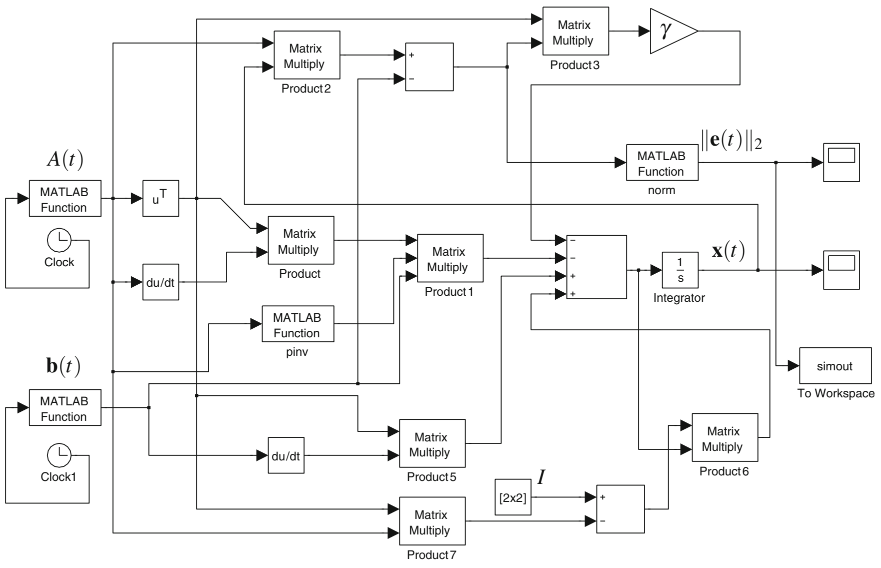


Fig. 5.4 Overall Simulink model of ZD (5.5) for solving over-determined system of time-varying linear equations

equations, i.e., (5.1) with $m < n$. Since $m < n$, there exist multiple or even an infinite number of solutions to (5.1).

Being similar to the above design procedure, in order to solve under-determined system of time-varying linear equations, we define the following two ZFs:

$$\mathbf{e}(t) = A(t)\mathbf{x}(t) - \mathbf{b}(t) \in \mathbb{R}^m, \tag{5.6}$$

$$\mathbf{e}(t) = \mathbf{x}(t) - A^{+}(t)\mathbf{b}(t) \in \mathbb{R}^n, \tag{5.7}$$

where $A^{+}(t) = A^{\mathrm{T}}(t)(A(t)A^{\mathrm{T}}(t))^{-1}$ denotes the right Moore–Penrose inverse of $A(t)$.

On the one hand, with ZF (5.6), by expanding the ZD design formula (4.2), we obtain the following ZD model for solving under-determined system of time-varying linear equations:

$$A(t)\dot{\mathbf{x}}(t) = -\dot{A}(t)\mathbf{x}(t) + \dot{\mathbf{b}}(t) - \gamma(A(t)\mathbf{x}(t) - \mathbf{b}(t)). \tag{5.8}$$

On the other hand, with ZF (5.7), by expanding the ZD design formula (4.2), we obtain another ZD model for solving under-determined system of time-varying linear equations as follows:

$$\dot{\mathbf{x}}(t) = -A^{+}(t)\dot{A}(t)A^{+}(t)\mathbf{b}(t) + A^{+}(t)\dot{\mathbf{b}}(t) - \gamma\left(\mathbf{x}(t) - A^{+}(t)\mathbf{b}(t)\right). \tag{5.9}$$

In summary, based on two different ZFs (5.6) and (5.7), two different ZD models (5.8) and (5.9) have been developed for solving under-determined system of time-varying linear equations, i.e., (5.1) with $m < n$. Note that the block diagrams and overall Simulink models corresponding to such two ZD models are left to interested readers to complete as a topic of exercise (since they are similar to those shown in Figs. 5.1, 5.2, 5.3, and 5.4).

5.3 Illustrative Examples

In this section, two illustrative examples are first simulated and analyzed for comparisons between the proposed ZD models (5.3) and (5.5) for solving over-determined system of time-varying linear equations. Then, another two illustrative examples are provided for substantiating the efficacy of the proposed ZD models (5.8) and (5.9) for solving under-determined system of time-varying linear equations.

Example 5.1 In the first example, the following smoothly time-varying coefficient matrix $A(t)$ and coefficient vector $\mathbf{b}(t)$ of (5.1) with $m = 3$ and $n = 2$ are designed to test the proposed ZD models (5.3) and (5.5):

$$A(t) = \begin{bmatrix} \sin(3t) & \cos(3t) \\ -\cos(3t) & \sin(3t) \\ \sin(3t) & \cos(3t) \end{bmatrix} \in \mathbb{R}^{3\times 2} \text{ and } \mathbf{b}(t) = \begin{bmatrix} \cos(t) \\ \sin(t) \\ -\sin(t) \end{bmatrix} \in \mathbb{R}^3.$$

The corresponding simulation results are shown in Figs. 5.5, 5.6, and 5.7.

Specifically, in the time period [0, 10]s, the state trajectories of the two elements $x_1(t)$ and $x_2(t)$ of $\mathbf{x}(t) = [x_1(t)\ x_2(t)]^{\mathrm{T}}$ synthesized by the proposed ZD models (5.3) and (5.5) with $\gamma = 1$ are illustrated in Fig. 5.5. It is seen that all simulated state trajectories (denoted by solid curves) starting from ten randomly-generated initial states $\mathbf{x}(0) \in [-1.5, 1.5]^2$ can relatively fast converge to the pseudoinverse-type solution $x^*(t) = A^+(t)b(t)$ which is exploited and shown for comparison and denoted by dash-dotted curves. Furthermore, Fig. 5.6 shows the residual errors $\|\mathbf{e}(t)\|_2 = \|A(t)\mathbf{x}(t) - \mathbf{b}(t)\|_2$ of ZD models (5.3) and (5.5) with $\gamma = 1$. As seen from Fig. 5.6, the residual errors of both ZD models cannot converge to zero. This phenomenon actually reflects and confirms that, for solving such an inconsistent

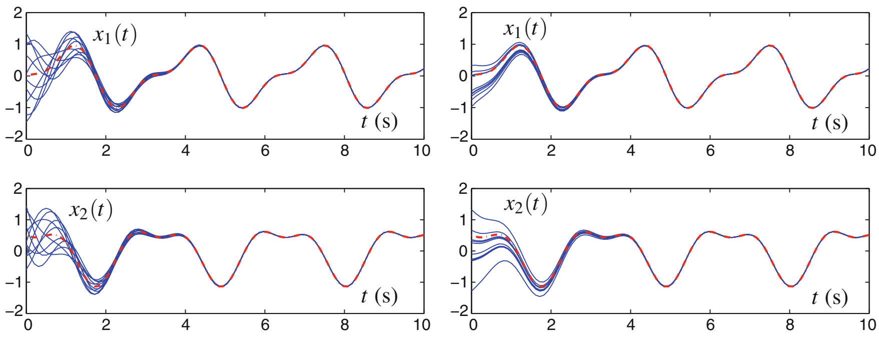


Fig. 5.5 State trajectories of ZD models (5.3) and (5.5) with $\gamma = 1$ for solving over-determined system of time-varying linear equations involved in Example 5.1, where the *dash-dotted* curves correspond to the pseudoinverse-type solution $\mathbf{x}^*(t) = A^+(t)\mathbf{b}(t)$

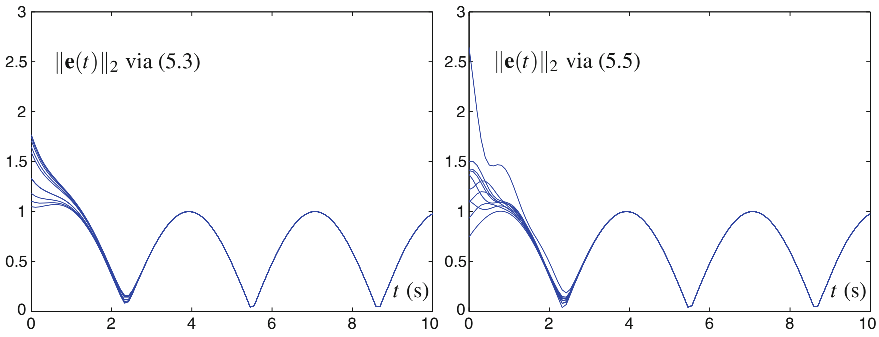


Fig. 5.6 Residual errors $\|\mathbf{e}(t)\|_2 = \|A(t)\mathbf{x}(t) - \mathbf{b}(t)\|_2$ of ZD models (5.3) and (5.5) with $\gamma = 1$ for solving over-determined system of time-varying linear equations involved in Example 5.1

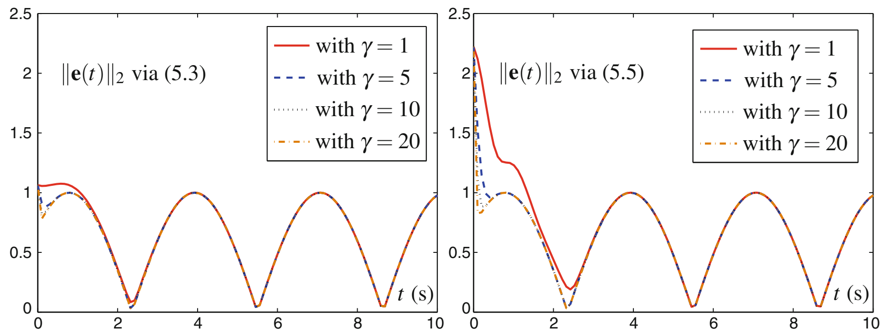


Fig. 5.7 Residual errors $\|\mathbf{e}(t)\|_2 = \|A(t)\mathbf{x}(t) - \mathbf{b}(t)\|_2$ of ZD models (5.3) and (5.5) using different γ values for solving over-determined system of time-varying linear equations involved in Example 5.1

over-determined system of time-varying linear equations, we cannot find a time-varying solution that satisfies all of the inconsistent equations simultaneously all the time.

Moreover, as seen from Fig. 5.7, ZD models (5.3) and (5.5) using different values of γ are investigated. Synthesized by the proposed ZD models, the residual errors (with $\gamma = 1$, 5, 10, and 20) cannot converge to zero either, and the reason has been explained in the preceding paragraph. Besides, as shown in Fig. 5.7, the residual errors with larger γ value converge faster than those with smaller γ value, showing that γ plays an important role in such ZD models.

Example 5.2 In the second example, the following time-varying coefficients of (5.1) with $m = 5$ and $n = 4$ are designed to test the proposed ZD models (5.3) and (5.5):

$$A(t) = \begin{bmatrix} a_1(t) & a_2(t) & a_3(t) & a_4(t) \\ a_1(t) & -a_2(t) & a_3(t) & a_4(t) \\ a_1(t) & a_2(t) & -a_3(t) & a_4(t) \\ a_1(t) & a_2(t) & a_3(t) & -a_4(t) \\ a_1(t) & a_2(t) & a_3(t) & a_4(t) \end{bmatrix} \in \mathbb{R}^{5\times 4} \text{ and } \mathbf{b}(t) = \begin{bmatrix} 2\sin(t) \\ 3\cos(2t) \\ 4\sin(2t) \\ 3\cos(t) \\ \sin(2t) \end{bmatrix} \in \mathbb{R}^5.$$

where $a_1(t) = 4 - \sin(t)$, $a_2(t) = 2 + \cos(2t)$, $a_3(t) = 3 - \sin(2t)$ and $a_4(t) = 2 + \cos(t)$. The corresponding simulation results are shown in Figs. 5.8, 5.9, and 5.10.

Specifically, in the time period [0, 10]s, the state trajectories of the four elements $x_1(t)$, $x_2(t)$, $x_3(t)$, and $x_4(t)$ of $\mathbf{x}(t) = [x_1(t)\ x_2(t)\ x_3(t)\ x_4(t)]^{\mathrm{T}}$ synthesized by ZD models (5.3) and (5.5) with $\gamma = 1$ are illustrated in Fig. 5.8. Starting from ten randomly-generated initial states $\mathbf{x}(0) \in [-1.5, 1.5]^4$, all state trajectories (denoted by solid curves) can also relatively fast converge to the pseudoinverse-type solution $\mathbf{x}^*(t) = A^+(t)\mathbf{b}(t)$ (denoted by dash-dotted curves again).

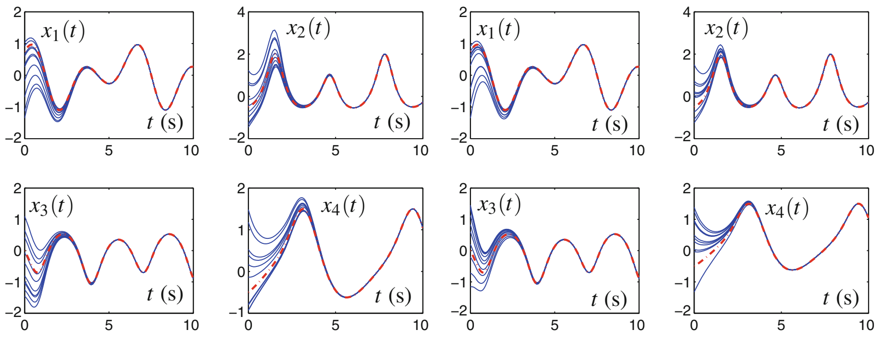


Fig. 5.8 State trajectories of ZD models (5.3) and (5.5) with $\gamma = 1$ for solving over-determined system of time-varying linear equations involved in Example 5.1

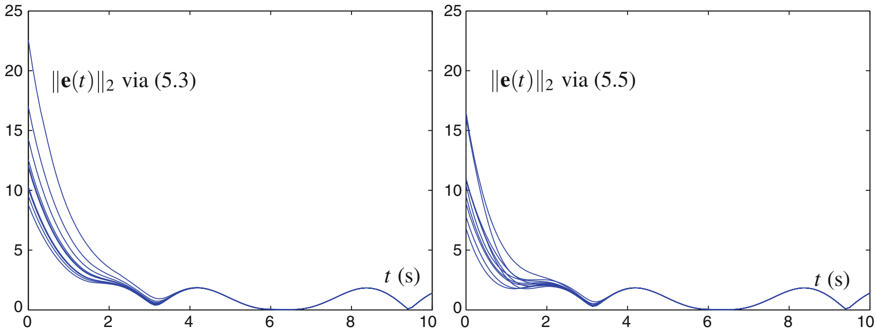


Fig. 5.9 Residual errors $\|\mathbf{e}(t)\|_2 = \|A(t)\mathbf{x}(t) - \mathbf{b}(t)\|_2$ of ZD models (5.3) and (5.5) with $\gamma = 1$ for solving time-varying over-determined system of linear equations involved in Example 5.1

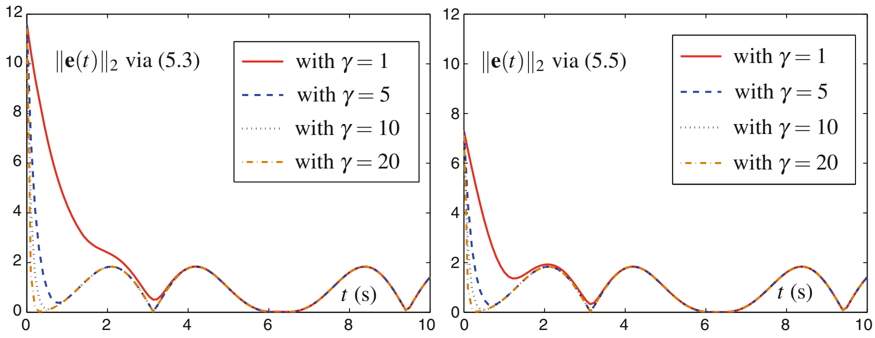


Fig. 5.10 Residual errors $\|\mathbf{e}(t)\|_2 = \|A(t)\mathbf{x}(t) - \mathbf{b}(t)\|_2$ of ZD models (5.3) and (5.5) using different γ values for solving over-determined system of time-varying linear equations involved in Example 5.2

Furthermore, Fig. 5.9 shows the residual errors $\|\mathbf{e}(t)\|_2 = \|A(t)\mathbf{x}(t) - \mathbf{b}(t)\|_2$ of ZD models (5.3) and (5.5) with $\gamma = 1$. As seen from the figure, the residual errors of ZD models (5.3) and (5.5) cannot converge to zero either. The reason is explained before; i.e., we cannot find a time-varying solution which can satisfy all of the inconsistent equations simultaneously.

Moreover, as seen from Fig. 5.10, ZD models (5.3) and (5.5) using different values of γ are investigated. We confirmedly observe that the residual errors with larger γ value converge faster than those with smaller γ value (showing again the important role of γ for the proposed ZD models).

In summary, the simulation results of the above two examples have substantiated the efficacy of the proposed ZD models (5.3) and (5.5) (derived from two different ZFs) for solving over-determined system of time-varying linear equations.

Example 5.3 In the third example, the following smoothly time-varying coefficient matrix $A(t)$ and coefficient vector $\mathbf{b}(t)$ of (5.1) with $m = 2$ and $n = 3$ are designed to test the proposed ZD models (5.8) and (5.9):

$$A(t) = \begin{bmatrix} \sin(0.6t) & \cos(0.6t) & -\sin(0.6t) \\ -\cos(0.6t) & \sin(0.6t) & \cos(0.6t) \end{bmatrix} \in \mathbb{R}^{2\times 3} \text{ and } \mathbf{b}(t) = \begin{bmatrix} 1.5\cos(t) \\ \sin(2t) \end{bmatrix} \in \mathbb{R}^2.$$

The corresponding simulation results are shown in Figs. 5.11, 5.12, and 5.13.

Specifically, in the time period $t \in [0, 10]$s, state trajectories of the elements $x_1(t)$, $x_2(t)$, and $x_3(t)$ (denoted by solid curves) synthesized by ZD models (5.8) and (5.9) with $\gamma = 1$ are illustrated in Fig. 5.11. Evidently, starting from ten randomly-generated initial states $\mathbf{x}(0) \in [-2, 2]^3$, some of the simulated state trajectories synthesized by ZD model (5.8) (e.g., $x_1(t)$ in the left graph of Fig. 5.11) do not converge to the trajectories of the referenced theoretical solution $\mathbf{x}^*(t) = A^+(t)\mathbf{b}(t)$ (denoted by dash-dotted curves), but run in parallel with the theoretical-solution trajectories. The reason is that there are multiple time-varying solutions satisfying

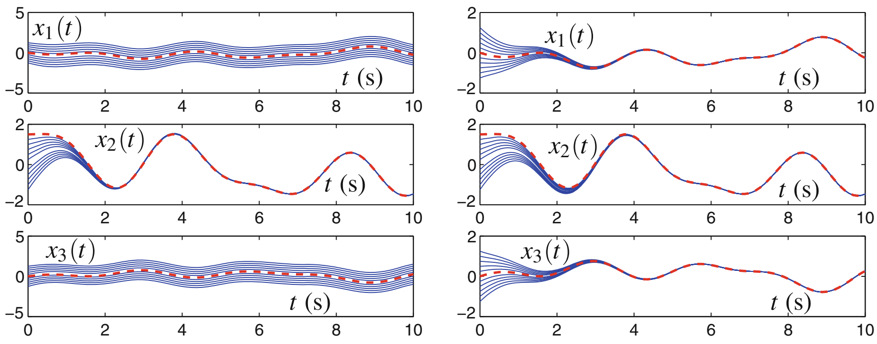


Fig. 5.11 State trajectories of ZD models (5.8) and (5.9) with $\gamma = 1$ for solving under-determined system of time-varying linear equations involved in Example 5.3

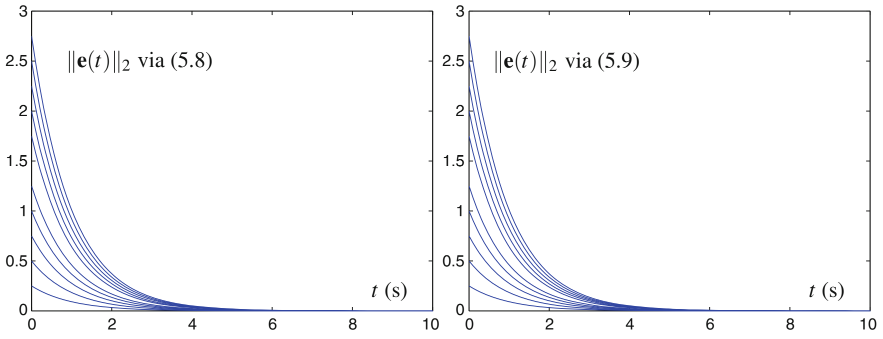


Fig. 5.12 Residual errors $\|\mathbf{e}(t)\|_2 = \|A(t)\mathbf{x}(t) - \mathbf{b}(t)\|_2$ of ZD models (5.8) and (5.9) with $\gamma = 1$ for solving under-determined system of time-varying linear equations involved in Example 5.3

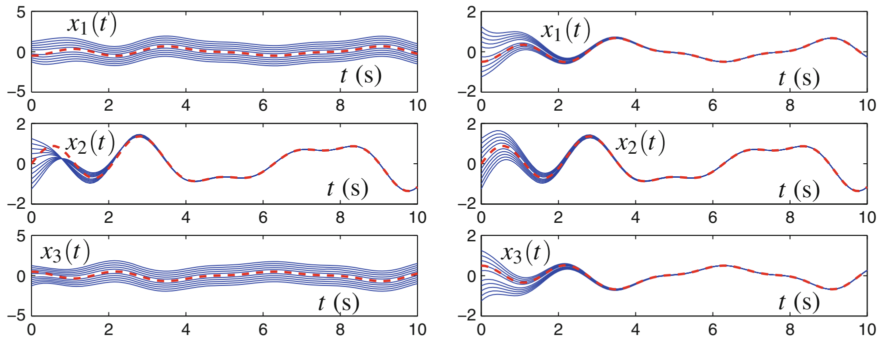


Fig. 5.13 State trajectories of ZD models (5.8) and (5.9) with $\gamma = 1$ for solving under-determined system of time-varying linear equations involved in Example 5.4

the under-determined system of time-varying linear equations with different initial states $\mathbf{x}(0)$ used. In contrast, other simulated state trajectories of ZD model (5.8) and all simulated state trajectories of ZD model (5.9), starting from randomly-generated initial states, relatively fast converge to the trajectories of the referenced theoretical solution $\mathbf{x}^*(t) = A^+(t)\mathbf{b}(t)$, as shown in Fig. 5.11.

Furthermore, Fig. 5.12 shows the residual errors $\|\mathbf{e}(t)\|_2 = \|A(t)\mathbf{x}(t) - \mathbf{b}(t)\|_2$ of ZD models (5.8) and (5.9) with $\gamma = 1$ used. As seen from the figure, the residual errors $\|\mathbf{e}(t)\|_2$ fast converge to zero. Note that the simulation results synthesized by ZD models (5.8) and (5.9) using different γ values are similar to those shown in Figs. 5.7 and 5.10 (and thus are omitted due to results similarity). That is, the residual errors with larger γ value converge faster than those with smaller γ value, showing the important role of γ for the proposed ZD models (5.8) and (5.9).

Example 5.4 In the fourth example, the following smoothly time-varying coefficient matrix $A(t)$ and coefficient vector $\mathbf{b}(t)$ of (5.1) with $m = 2$ and $n = 3$ are designed to test the proposed ZD models (5.8) and (5.9):

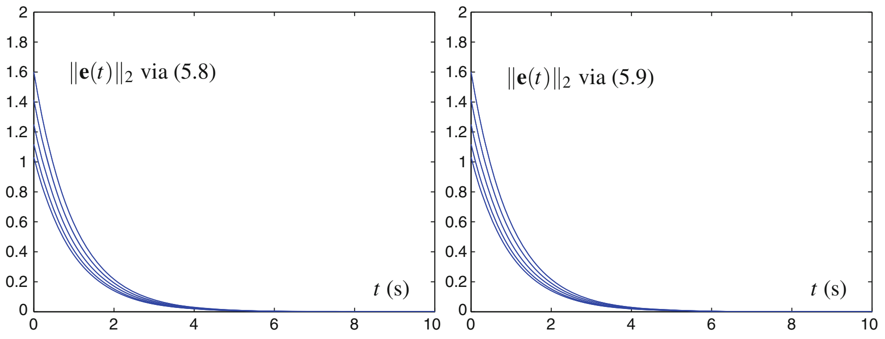


Fig. 5.14 Residual errors $\|\mathbf{e}(t)\|_2 = \|A(t)\mathbf{x}(t) - \mathbf{b}(t)\|_2$ of ZD models (5.8) and (5.9) with $\gamma = 1$ for solving under-determined system of time-varying linear equations involved in Example 5.4

$$A(t) = \begin{bmatrix} \sin(2t) & \cos(2t) & -\sin(2t) \\ -\cos(2t) & \sin(2t) & \cos(2t) \end{bmatrix} \in \mathbb{R}^{2\times 3} \text{ and } \mathbf{b}(t) = \begin{bmatrix} \sin(0.5t) \\ \cos(t) \end{bmatrix} \in \mathbb{R}^2.$$

The corresponding simulation results are shown in Figs. 5.13 and 5.14, where phenomena are similar to those in Example 5.3. That is, corresponding to $\mathbf{x}(t)$ in Fig. 5.13, the residual errors of ZD models (5.8) and (5.9) in Fig. 5.14 all converge to zero. Note that, being a topic of exercise, the related simulative verifications of ZD models (5.8) and (5.9) using different values of γ are left for interested readers.

In summary, the simulation results of the above two illustrative examples have substantiated the efficacy of the proposed ZD models (5.8) and (5.9) for solving under-determined system of time-varying linear equations.

5.4 Summary

In this chapter, by introducing different ZFs [i.e., (5.2), (5.4), (5.6), and (5.7)], different ZD models [i.e., (5.3), (5.5), (5.8), and (5.9)] have been proposed, generalized, developed, and investigated to solve over-determined and under-determined systems of time-varying linear equations (5.1). With different illustrative examples, computer simulation results have further substantiated the efficacy of the proposed ZD models for solving over-determined and under-determined systems of time-varying linear equations.

References

1. Siciliano B, Khatib O (2008) Springer handbook of robotics. Springer, Heidelberg
2. Siciliano B, Sciavicco L, Villani L, Oriolo G (2009) Robotics: modelling, planning and control. Springer, London
3. Zhang Y, Zhang Z (2013) Repetitive motion planning and control of redundant robot manipulators. Springer, New York
4. Brannigan M (1982) The strict Chebyshev solution of overdetermined systems of linear equations with rank deficient matrix. Numer Math 40(3):307–318
5. Grava T (2002) Riemann-Hilbert problem for the small dispersion limit of the KdV equation and linear overdetermined systems of Euler-Poisson-Darboux type. Commun Pure Appl Math 55(4):395–430
6. Cadzow JA (2002) Minimum ℓ_1, ℓ_2, and ℓ_∞ norm approximate solutions to an overdetermined system of linear equations. Dig Signal Process 12(4):524–560
7. Godunov SK (2008) On approximations for overdetermined hyperbolic equations. In: Proceedings of the 12th international conference on hyperbolic problems: theory, numerics, applications, pp 19–33
8. Rosen JB, Park H, Glick J, Zhang L (2000) Accurate solution to overdetermined linear equations with errors using L_1 norm minimization. Comput Optim Appl 17(2–3):329–341
9. Rathinavelan T, Wonpil IM (2007) Explicit treatment of force contribution from alignment tensor using overdetermined linear equations and its application in NMR structure determination. J Comput Chem 28(11):1858–1864
10. Zhang Y, Li W, Guo D, Mu B, Zheng H (2013) Different Zhang functions leading to various ZNN models illustrated via solving the time-varying overdetermined system of linear equations. In: Proceedings of international conference on information science and technology, pp 771–776
11. Zhang Y, Wang Y, Jin L, Mu B, Zheng H (2013) Different ZFs leading to various ZNN models illustrated via online solution of time-varying underdetermined systems of linear equations with robotic application. Lect Notes Comput Sci 7952:481–488
12. Mathews JH, Fink KD (2004) Numerical methods using MATLAB. Prentice Hall, New Jersey

Chapter 6
Time-Varying Linear Matrix-Vector Inequality

Abstract In this chapter, by defining three different ZFs, three different ZD models are proposed, generalized, developed, and investigated to solve the time-varying linear matrix-vector inequality. Theoretical results are given as well to show the excellent convergence performance of such ZD models. Computer simulation results are presented to further substantiate the efficacy of the proposed ZD models for time-varying linear matrix-vector inequality solving.

6.1 Introduction

Online solution of linear matrix-vector inequality in the form of $A\mathbf{x} \leqslant \mathbf{b}$ (where A denotes a constant matrix and $\mathbf{b}$ denotes a constant vector) and time-varying linear matrix-vector inequality in the form of $A(t)\mathbf{x}(t) \leqslant \mathbf{b}(t)$ [where $A(t)$ denotes a time-varying matrix and $\mathbf{b}(t)$ denotes a time-varying vector] is considered to be an important issue encountered in science and engineering fields [1–10], e.g., image restoration [1], digital signal processing [4], and robot inverse kinematics [7]. Specifically, time-varying linear matrix-vector inequality has a wide application in motion planning of robot manipulators [8–10]; e.g., different time-varying linear matrix-vector inequalities are introduced and investigated for avoiding obstacles [9, 10], which are used to generate escape velocities of variable magnitude, driving the affected links away from obstacles. Thus, robot manipulators can avoid obstacles successfully when approaching them. Due to the in-depth researches, a variety of approaches (including numerical algorithms and neural networks) for solving (time-varying and/or time-invariant) linear matrix-vector inequality have been developed and investigated [2–6, 11–15].

In this chapter, focusing on solving time-varying linear matrix-vector inequality, we propose, generalize, develop, and investigate three different ZD models by defining three different ZFs. As for such ZD models, theoretical results are given as well to show their excellent convergence performance. Two illustrative examples are provided and computer simulation results further substantiate the efficacy of the proposed ZD models for time-varying linear matrix-vector inequality solving.

Y. Zhang and D. Guo, *Zhang Functions and Various Models*,
DOI 10.1007/978-3-662-47334-4_6

6.2 ZFs and ZD Models

In this section, by defining different ZFs, different ZD models are proposed for solving the following time-varying linear matrix-vector inequality:

$$A(t)\mathbf{x}(t) \leqslant \mathbf{b}(t), \ t \in [0, +\infty), \tag{6.1}$$

where $A(t) \in \mathbb{R}^{n\times n}$ is the smoothly time-varying nonsingular coefficient matrix, $\mathbf{b}(t) \in \mathbb{R}^n$ is the smoothly time-varying coefficient vector, and $\mathbf{x}(t) \in \mathbb{R}^n$ is the unknown vector that needs to be obtained.

To lay a basis for further discussion, let us define the theoretical time-varying solution set $\mathbb{S}(t) = \{\mathbf{x}(t)|\mathbf{x}(t) \in \mathbb{R}^n$ is a solution of 6.1$\}$ with $\mathbb{S}(0)$ denoting its initial solution set. For presentation convenience, we define the residual error as

$$\mathbf{y}(t) = A(t)\mathbf{x}(t) - \mathbf{b}(t) \in \mathbb{R}^n,$$

and $\mathbf{y}(t) = [y_1(t), y_2(t), \ldots, y_n(t)]^{\mathrm{T}}$. In addition, we use $\max\{\mathbf{0}, \mathbf{y}(t)\}$ as a criterion to measure the efficacy of ZD models for online solution of time-varying linear matrix-vector inequality. This is because if $\max\{\mathbf{0}, \mathbf{y}(t)\} = \mathbf{0}$ holds true, then $\mathbf{y}(t) = A(t)\mathbf{x}(t) - \mathbf{b}(t) \leqslant \mathbf{0}$ and we can say that the resultant time-varying solution $x(t)$ is the desired solution of (6.1). This paper aims at defining different ZFs to result in different types of ZD models for solving time-varying linear matrix-vector inequality (6.1) and finding an unknown $\mathbf{x}(t) \in \mathbb{S}(t)$ in real time t such that $\max\{\mathbf{0}, \mathbf{y}(t)\} \to \mathbf{0}$ as time t goes on.

6.2.1 The First ZF and ZD Model

In order to solve time-varying linear matrix-vector inequality (6.1), the first ZF is defined as follows:

$$\mathbf{e}(t) = \begin{bmatrix} e_1(t) \\ e_2(t) \\ \vdots \\ e_n(t) \end{bmatrix} = \begin{bmatrix} (\max\{0, y_1(t)\})^2/2 \\ (\max\{0, y_2(t)\})^2/2 \\ \vdots \\ (\max\{0, y_n(t)\})^2/2 \end{bmatrix} \in \mathbb{R}^n. \tag{6.2}$$

Based on ZF (6.2), by expanding the ZD design formula (4.2), we obtain

$$A(t)\dot{\mathbf{x}}(t) = -\dot{A}(t)\mathbf{x}(t) + \dot{\mathbf{b}}(t) - \frac{\gamma}{2}\max\{\mathbf{0}, A(t)\mathbf{x}(t) - \mathbf{b}(t)\} \tag{6.3}$$

which is the first ZD model for time-varying linear matrix-vector inequality solving.

As for ZD model (6.3), we have the following important theorem to guarantee that, as $t \to \infty$, $\|\mathbf{e}(t)\|_2 \to 0$ globally and exponentially. Note that the corresponding proof is given in [13] and thus is not repeated again (but left to interested readers to complete as a topic of exercise).

Theorem 6.1 *For ZD model (6.3), given a smoothly time-varying nonsingular coefficient matrix $A(t)$ and a smoothly time-varying coefficient vector $\mathbf{b}(t)$ in (6.1),*

- *if initial state $\mathbf{x}(0) \in \mathbb{R}^n$ is outside the initial solution set $\mathbb{S}(0)$ of (6.1), the norm-based error function $\|\mathbf{e}(t))\|_2$ is globally and exponentially convergent to zero with convergence rate γ;*
- *if initial state $\mathbf{x}(0) \in \mathbb{R}^n$ is inside the initial solution set $\mathbb{S}(0)$ of (6.1), the norm-based error function $\|\mathbf{e}(t)\|_2$ is always equal to zero.*

That is, ZD model (6.3) generates an exact time-varying solution of (6.1), $A(t)\mathbf{x}(t) \leqslant \mathbf{b}(t)$, with an exponential convergence performance.

6.2.2 The Second ZF and ZD Model

In this subsection, being different from the first ZF (6.2), the second ZF is defined as below:

$$\mathbf{e}(t) = \begin{bmatrix} e_1(t) \\ e_2(t) \\ \vdots \\ e_n(t) \end{bmatrix} = \begin{bmatrix} y_1(t) \\ y_2(t) \\ \vdots \\ y_n(t) \end{bmatrix} = \mathbf{y}(t) = A(t)\mathbf{x}(t) - \mathbf{b}(t) \in \mathbb{R}^n. \tag{6.4}$$

Then, in order to obtain an effective ZD model for solving time-varying linear matrix-vector inequality (6.1), a new design formula is elaborately constructed for the second ZF (6.4) as follows [which is clearly different from the design formula (4.2) presented in Chap. 4]:

$$\dot{\mathbf{e}}(t) = \frac{\mathrm{d}\mathbf{e}(t)}{\mathrm{d}t} = -\gamma \mathrm{JMP}\left(\mathbf{e}(0)\right) \diamond \mathbf{e}(t), \tag{6.5}$$

where $\mathrm{JMP}(\cdot) : \mathbb{R}^n \to \mathbb{R}^n$ denotes an array of jump functions with each element defined as

$$\mathrm{jmp}(c) = \begin{cases} 1, & \text{if } \ c > 0, \\ 0, & \text{if } \ c \leqslant 0. \end{cases}$$

In addition, the vector multiplication operator $\diamond$ is defined as

$$\mathbf{u} \diamond \mathbf{v} = \begin{bmatrix} u_1 v_1 \\ u_2 v_2 \\ \vdots \\ u_n v_n \end{bmatrix} \in \mathbb{R}^n \text{ with } \mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} \in \mathbb{R}^n \text{ and } \mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} \in \mathbb{R}^n.$$

In view of ZF (6.4) and $\dot{\mathbf{y}}(t) = \dot{A}(t)\mathbf{x}(t) + A(t)\dot{\mathbf{x}}(t) - \dot{\mathbf{b}}(t)$, by expanding the ZD design formula (6.5), we obtain the following ZD model for time-varying linear matrix-vector inequality solving:

$$A(t)\dot{\mathbf{x}}(t) = -\gamma \mathrm{JMP}\,(A(0)\mathbf{x}(0) - \mathbf{b}(0)) \diamond (A(t)\mathbf{x}(t) - \mathbf{b}(t)) - \dot{A}(t)\mathbf{x}(t) + \dot{\mathbf{b}}(t). \quad (6.6)$$

As for ZD model (6.6), we have the following important theorem to guarantee the convergence performance of such a model. Note that the corresponding proof is generalized from the proof shown in [6], and is also left to interested readers to complete as a topic of exercise).

Theorem 6.2 *Given a smoothly time-varying nonsingular coefficient matrix $A(t)$ and a smoothly time-varying coefficient vector $\mathbf{b}(t)$ in (6.1), for ZD model (6.6),*

- *if initial state $\mathbf{x}(0) \in \mathbb{R}^n$ is outside the initial solution set $\mathbb{S}(0)$ of (6.1), then there exists at least one $i \in \{1, 2, \ldots, n\}$ such that $e_i(t)$ is globally and exponentially convergent to zero with convergence rate λ, while the other error functions $e_j(t)$, with $j = 1, 2, \ldots, n$ and $j \neq i$, are always equal to $e_j(0)$;*
- *if initial state $\mathbf{x}(0) \in \mathbb{R}^n$ is inside the initial solution set $\mathbb{S}(0)$ of (6.1), the error function $\mathbf{e}(t)$ is always equal to $\mathbf{e}(0)$.*

That is, ZD model (6.6) generates an exact time-varying solution of (6.1) with an exponential convergence performance.

6.2.3 The Third ZF and ZD Model

In this subsection, we aim at developing another effective ZD model to solve time-varying linear matrix-vector inequality (6.1). Inspired by [16, 17], we surprisedly discover that solving linear matrix-vector inequality (6.1) can be equivalently converted to linear matrix-vector equation solving. Therefore, an equivalent time-varying matrix-vector equation is firstly derived from a time-varying matrix-vector inequality by introducing a time-varying nonnegative vector. Then, by defining the third ZF, the corresponding ZD model is developed and investigated for solving time-varying linear matrix-vector inequality (6.1).

Conversion In order to convert time-varying linear matrix-vector inequality (6.1) to a time-varying matrix-vector equation, a time-varying nonnegative vector is introduced and incorporated into time-varying linear matrix-vector inequality (6.1). Specifically, (6.1) can be transformed into

$$A(t)\mathbf{x}(t) - \mathbf{b}(t) + \tilde{\mathbf{x}}^2(t) = \mathbf{0} \in \mathbb{R}^n, \tag{6.7}$$

where parameter factor $\tilde{\mathbf{x}}^2(t) := \tilde{\mathbf{x}}(t) \diamond \tilde{\mathbf{x}}(t)$ with the time-varying vector $\tilde{\mathbf{x}}(t) = [\tilde{x}_1(t), \tilde{x}_2(t), \ldots, \tilde{x}_n(t)]^{\mathrm{T}} \in \mathbb{R}^n$. Evidently, $\tilde{\mathbf{x}}^2(t) \geqslant \mathbf{0}$, and thus time-varying matrix-vector equation (6.7) is equivalent to time-varying linear matrix-vector inequality (6.1) in view of

$$A(t)\mathbf{x}(t) - \mathbf{b}(t) = -\tilde{\mathbf{x}}^2(t) \leqslant \mathbf{0}.$$

Therefore, for solving time-varying linear matrix-vector inequality (6.1), we only need to solve such a time-varying linear matrix-vector equation (6.7). Then, for the need of modeling, we define the diagonal matrix $C(t)$ as

$$C(t) = \begin{bmatrix} \tilde{x}_1(t) & 0 & \cdots & 0 \\ 0 & \tilde{x}_2(t) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \tilde{x}_n(t) \end{bmatrix} \in \mathbb{R}^{n\times n}.$$

Thus, one can obtain

$$\tilde{\mathbf{x}}^2(t) = C(t)\tilde{\mathbf{x}}(t), \text{ and } \dot{\tilde{\mathbf{x}}}^2(t) = \frac{\mathrm{d}\tilde{\mathbf{x}}^2(t)}{\mathrm{d}t} = 2C(t)\dot{\tilde{\mathbf{x}}}(t).$$

Note that the above equations would be used to obtain the third ZD model.

Model Formulation To monitor and control the process of solving time-varying linear matrix-vector equation (6.7) and time-varying linear matrix-vector inequality (6.1), we define the third ZF as follows:

$$\mathbf{e}(t) = \begin{bmatrix} e_1(t) \\ e_2(t) \\ \vdots \\ e_n(t) \end{bmatrix} = \begin{bmatrix} y_1(t) + \tilde{x}_1^2(t) \\ y_2(t) + \tilde{x}_2^2(t) \\ \vdots \\ y_n(t) + \tilde{x}_n^2(t) \end{bmatrix} = \mathbf{y}(t) + \tilde{\mathbf{x}}^2(t) \in \mathbb{R}^n. \tag{6.8}$$

With ZF (6.8), by expanding the ZD design formula (4.2), we obtain the following dynamic equation of a ZD model:

$$A(t)\dot{\mathbf{x}}(t) + 2C(t)\dot{\tilde{\mathbf{x}}}(t) = -\dot{A}(t)\mathbf{x}(t) + \dot{\mathbf{b}}(t) - \gamma(A(t)\mathbf{x}(t) - \mathbf{b}(t) + C(t)\tilde{\mathbf{x}}(t)). \tag{6.9}$$

Let $\mathbf{w}(t) = [\mathbf{x}^{\mathrm{T}}(t), \tilde{\mathbf{x}}^{\mathrm{T}}(t)]^{\mathrm{T}} \in \mathbb{R}^{2n}$, and then (6.9) is reformulated as below:

$$P(t)\dot{\mathbf{w}}(t) = G(t)\mathbf{w}(t) + \dot{\mathbf{b}}(t) - \gamma(D(t)\mathbf{w}(t) - \mathbf{b}(t)), \tag{6.10}$$

where the augmented matrices are defined as follows:

$$P(t) = \begin{bmatrix} A^{\mathrm{T}}(t) \\ 2C^{\mathrm{T}}(t) \end{bmatrix}^{\mathrm{T}}, \; G(t) = \begin{bmatrix} -\dot{A}^{\mathrm{T}}(t) \\ 0 \end{bmatrix}^{\mathrm{T}}, \text{ and } D(t) = \begin{bmatrix} A^{\mathrm{T}}(t) \\ C^{\mathrm{T}}(t) \end{bmatrix}^{\mathrm{T}}.$$

Thus, based on ZF (6.8), ZD model (6.10) is obtained for solving time-varying linear matrix-vector equation (6.7) and time-varying linear matrix-vector inequality (6.1). Besides, the block diagram of ZD model (6.10) is shown in Fig. 6.1, which is an important and necessary step for the final hardware implementation.

Theoretical Results and Analysis In this part, we come to prove the convergence performance of ZD model (6.10) through following important theorem.

Theorem 6.3 *Given a smoothly time-varying nonsingular coefficient matrix $A(t)$ and a smoothly time-varying coefficient vector $\mathbf{b}(t)$ in (6.1), ZD model (6.10), starting from any randomly-generated initial state, converges exponentially to the time-varying solution of time-varying linear matrix-vector equation (6.7) with rate γ, of which the first n elements constitute an exact time-varying solution of time-varying linear matrix-vector inequality (6.1).*

Proof Consider time-varying linear matrix-vector equation (6.7), i.e., $A(t)\mathbf{x}(t) - \mathbf{b}(t) + \tilde{\mathbf{x}}^2(t) = \mathbf{0}$, which can be rewritten as

$$D(t)\mathbf{w}(t) - \mathbf{b}(t) = \mathbf{0}.$$

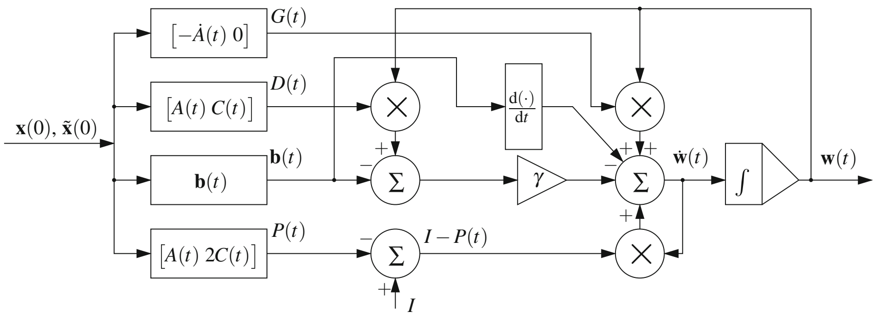


Fig. 6.1 Block diagram of ZD model (6.10) for time-varying linear matrix-vector inequality (6.1) solving

Evidently, ZD model (6.10) is derived from the following design formula:

$$\frac{\mathrm{d}(D(t)\mathbf{w}(t)-\mathbf{b}(t))}{\mathrm{d}t}=-\gamma(D(t)\mathbf{w}(t)-\mathbf{b}(t)).$$

From the above equation, we obtain

$$D(t)\mathbf{w}(t)-\mathbf{b}(t)=[D(0)\mathbf{w}(0)-\mathbf{b}(0)]\exp(-\gamma t).$$

which means that, as $t\to\infty$, $D(t)\mathbf{w}(t)-\mathbf{b}(t)\to\mathbf{0}$ globally and exponentially. That is, ZD model (6.10), starting from any randomly-generated initial state $\mathbf{w}(0)$, converges exponentially to the time-varying solution of time-varying linear matrix-vector equation (6.7) with rate γ. The first part is thus completed. Next, we are going to prove the other part.

Let $\mathbf{x}^*(t)$ be a theoretical solution of (6.1), i.e., $A(t)\mathbf{x}^*(t)-\mathbf{b}(t)\leqslant\mathbf{0}$. Thus, there exists a time-varying nonnegative vector $\tilde{\mathbf{x}}^{*2}(t)\geqslant\mathbf{0}$, which makes the following time-varying matrix-vector equation hold true:

$$A(t)\mathbf{x}^*(t)+\tilde{\mathbf{x}}^{*2}(t)=\mathbf{b}(t). \tag{6.11}$$

Differentiating (6.11) with respect to time t, we can obtain

$$\dot{A}(t)\mathbf{x}^*(t)+A(t)\dot{\mathbf{x}}^*(t)+\dot{\tilde{\mathbf{x}}}^{*2}(t)=\dot{\mathbf{b}}(t), \tag{6.12}$$

where $\dot{\mathbf{x}}^*(t)$ and $\dot{\tilde{\mathbf{x}}}^{*2}(t)$ denote time derivatives of $\mathbf{x}^*(t)$ and $\tilde{\mathbf{x}}^{*2}(t)$, respectively. Then, substituting (6.11) and (6.12) into (6.10), we can obtain

$$\begin{aligned}A(t)(\dot{\mathbf{x}}(t)\ &-\dot{\mathbf{x}}^*(t))+\dot{A}(t)(\mathbf{x}(t)-\mathbf{x}^*(t))+\dot{\tilde{\mathbf{x}}}^{2}(t)-\dot{\tilde{\mathbf{x}}}^{*2}(t)\\&=-\gamma(A(t)(\mathbf{x}(t)-\mathbf{x}^*(t))+\tilde{\mathbf{x}}^{2}(t)-\tilde{\mathbf{x}}^{*2}(t)).\end{aligned}$$

Let $\tilde{\mathbf{e}}(t)=A(t)(\mathbf{x}(t)-\mathbf{x}^*(t))+\tilde{\mathbf{x}}^{2}(t)-\tilde{\mathbf{x}}^{*2}(t)$. Then, the above equation is rewritten as

$$\dot{\tilde{\mathbf{e}}}(t)=-\gamma\tilde{\mathbf{e}}(t). \tag{6.13}$$

Evidently, from (6.13), we can obtain

$$\tilde{\mathbf{e}}(t)=\tilde{\mathbf{e}}(0)\exp(-\gamma t),$$

which means that, as $t\to\infty$, $\tilde{\mathbf{e}}(t)\to\mathbf{0}$ globally and exponentially. In addition, based on (6.11), i.e., $\mathbf{b}(t)=A(t)\mathbf{x}^*(t)+\tilde{\mathbf{x}}^{*2}(t)$, $\tilde{\mathbf{e}}(t)$ is further reformulated as

$$\tilde{\mathbf{e}}(t)=A(t)\mathbf{x}(t)-\mathbf{b}(t)+\tilde{\mathbf{x}}^{2}(t),$$

Table 6.1 Different ZFs resulting in different ZD models for time-varying linear matrix-vector inequality solving

ZF	ZD model
(6.2)	$A(t)\dot{\mathbf{x}}(t) = -\gamma \max\{\mathbf{0}, A(t)\mathbf{x}(t) - \mathbf{b}(t)\}/2 - \dot{A}(t)\mathbf{x}(t) + \dot{\mathbf{b}}(t)$
(6.4)	$A(t)\dot{\mathbf{x}}(t) = -\gamma \mathrm{JMP}(A(0)\mathbf{x}(0) - \mathbf{b}(0)) \diamond (A(t)\mathbf{x}(t) - \mathbf{b}(t)) - \dot{A}(t)\mathbf{x}(t) + \dot{\mathbf{b}}(t)$
(6.8)	$P(t)\dot{\mathbf{w}}(t) = G(t)\mathbf{w}(t) + \dot{\mathbf{b}}(t) - \gamma(D(t)\mathbf{w}(t) - \mathbf{b}(t))$

which is equivalent to

$$A(t)\mathbf{x}(t) - \mathbf{b}(t) = -\tilde{\mathbf{x}}^2(t) + \tilde{\mathbf{e}}(t).$$

Therefore, $A(t)\mathbf{x}(t) - \mathbf{b}(t) \to -\tilde{\mathbf{x}}^2(t)$ globally and exponentially as $\tilde{\mathbf{e}}(t) \to \mathbf{0}$. Note that $\tilde{\mathbf{x}}^2(t)$ is an introduced time-varying nonnegative vector, and thus $-\tilde{\mathbf{x}}^2(t) \leqslant \mathbf{0}$. This implies that $\mathbf{x}(t)$ [being the first n element of $\mathbf{w}(t)$ in ZD model (6.10)] would exponentially converge to an exact time-varying solution of (6.1) with convergence rate γ. The proof is thus complete. □

In summary, three different ZFs [i.e., (6.2), (6.4) and (6.8)] are presented, which result in different ZD models [i.e., (6.3), (6.6) and (6.10)] for solving time-varying linear matrix-vector inequality (6.1). For readers' convenience and also for comparison, such three different ZD models are listed comparatively in Table 6.1.

6.3 Illustrative Examples

In the previous section, three different ZFs are presented, which result in different ZD models for solving time-varying linear matrix-vector inequality (6.1). Note that the first two ZD models shown in Table 6.1 have been investigated in [6, 13]. Thus, we only summarize and compare their design processes and final models for the completeness of this book. Besides, by introducing a time-varying nonnegative vector, time-varying linear matrix-vector inequality (6.1) is transformed equivalently into time-varying linear matrix-vector equation (6.7). In addition to detailed design process of the ZD models, the excellent convergence performance is analyzed. In this section, we further focus on the study of ZD model (6.10), and thus two illustrative examples are provided for substantiation of the efficacy of such a ZD model.

Example 6.1 Now let us consider linear matrix-vector inequality (6.1) with time-varying coefficients being as follows:

$$A(t) = \begin{bmatrix} 2\sin(10t)+4 & 3\cos(12t) \\ -\cos(12t) & \sin(10t)+5 \end{bmatrix} \in \mathbb{R}^{2\times 2} \text{ and } \mathbf{b}(t) = \begin{bmatrix} \sin(15t) \\ \cos(10t) \end{bmatrix} \in \mathbb{R}^2.$$

Considering that different initial states of the ZD model (6.10) may result in different convergent performance, we investigate the following two cases.

Case 1: Initial State Outside $\mathbb{S}(0)$ If initial state $\mathbf{x}(0)$ is outside the initial solution set $\mathbb{S}(0)$ of (6.1), applying ZD model (6.10) to solving time-varying linear matrix-vector equation (6.7) with $\gamma = 5$ and a randomly-generated initial value $\tilde{\mathbf{x}}(0)$, the dynamic performance of $\mathbf{x}(t)$ and $\tilde{\mathbf{x}}(t)$ is seen from Fig. 6.2. For a better understanding, Fig. 6.3 shows the transient behavior of $\|A(t)\mathbf{x}(t) - \mathbf{b}(t) + \tilde{\mathbf{x}}^2(t)\|_2$ synthesized by ZD model (6.10) for online solution of time-varying linear matrix-vector equation (6.7). It is seen from Fig. 6.3 that $\|A(t)\mathbf{x}(t) - \mathbf{b}(t) + \tilde{\mathbf{x}}^2(t)\|_2$ decreases to zero within 1.5 s. That is to say, $\mathbf{x}(t)$ and $\tilde{\mathbf{x}}(t)$ shown in Fig. 6.2 are a group of exact time-varying solution of time-varying linear matrix-vector equation (6.7).

For demonstrating the effectiveness of ZD model (6.10) on solving time-varying linear matrix-vector inequality (6.1), the transient behavior of $\max\{\mathbf{0}, \mathbf{y}(t)\}$ is shown in Fig. 6.3. It follows that $\mathbf{y}(t) = A(t)\mathbf{x}(t) - \mathbf{b}(t)$ is less than or equal to zero within 0.6 s. That is, $\mathbf{x}(t)$ shown in Fig. 6.2 is also an exact time-varying solution of time-

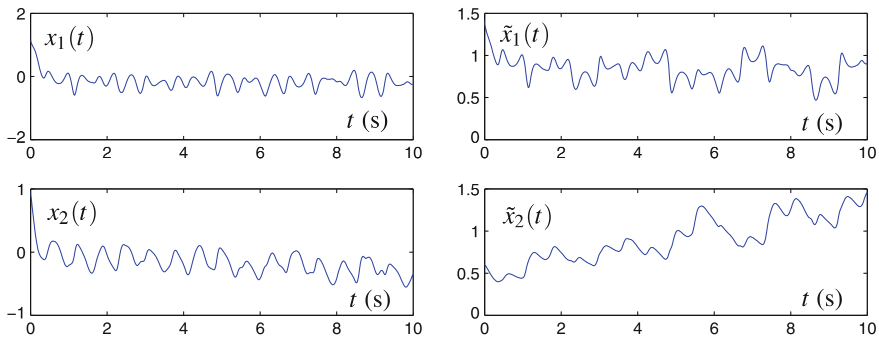


Fig. 6.2 Transient behavior of state vectors $\mathbf{x}(t)$ and $\tilde{\mathbf{x}}(t)$ synthesized by ZD model (6.10) starting from $\mathbf{x}(0) \notin \mathbb{S}(0)$ in Example 6.1

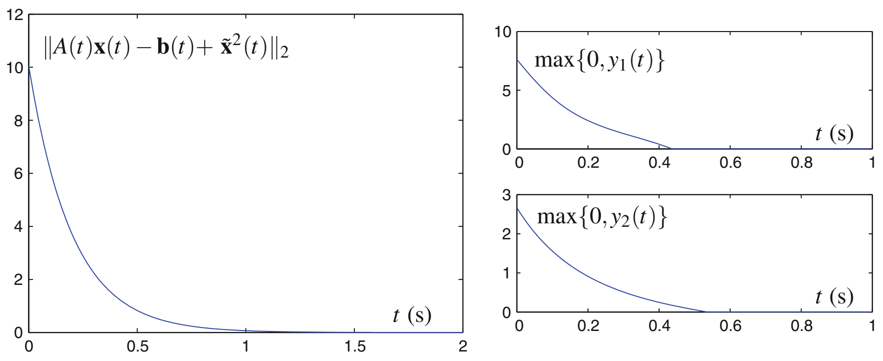


Fig. 6.3 Simulation results via ZD model (6.10) starting from $\mathbf{x}(0) \notin \mathbb{S}(0)$ in Example 6.1

varying linear matrix-vector inequality (6.1). The results demonstrate theoretical analysis of Theorem 6.3.

Case 2: Initial State Inside $\mathbb{S}(0)$ If initial state $\mathbf{x}(0)$ is inside the initial solution set $\mathbb{S}(0)$ of (6.1), under the same conditions, we apply ZD model (6.10) to solve time-varying linear matrix-vector inequality (6.1) and the corresponding time-varying linear matrix-vector equation (6.7). The corresponding simulation results are shown in Figs. 6.4 and 6.5.

Figure 6.4 shows the resultant solutions of time-varying linear matrix-vector equation (6.7), of which the left graph constitutes the solution of time-varying linear matrix-vector inequality (6.1). Similar with those of Fig. 6.2, $\mathbf{x}(t)$ and $\tilde{\mathbf{x}}(t)$ also possess the dynamic performance (i.e., time-varying performance). In addition, as seen from Fig. 6.5, $\|A(t)\mathbf{x}(t) - \mathbf{b}(t) + \tilde{\mathbf{x}}^2(t)\|_2$ synthesized by ZD model (6.10) converges to zero within 1.5 s as well. That is, ZD model (6.10) is still effective on solving time-varying linear matrix-vector equation (6.7) when initial state $\mathbf{x}(0)$ is inside the initial solution set $\mathbb{S}(0)$ of (6.1) and $\tilde{\mathbf{x}}(0)$ is a randomly-generated initial value.

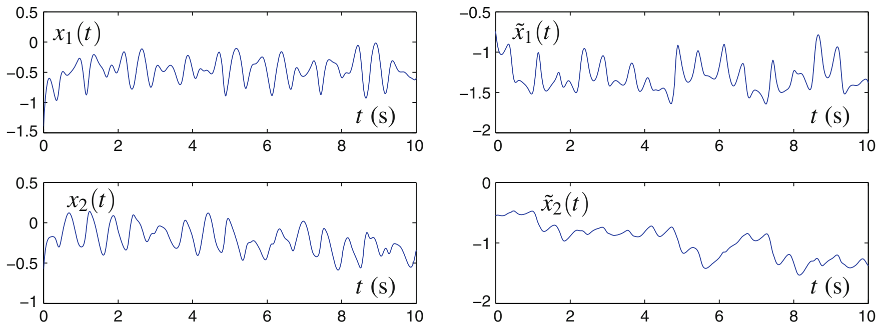


Fig. 6.4 Transient behavior of state vectors $\mathbf{x}(t)$ and $\tilde{\mathbf{x}}(t)$ synthesized by ZD model (6.10) starting from $\mathbf{x}(0) \in \mathbb{S}(0)$ in Example 6.1

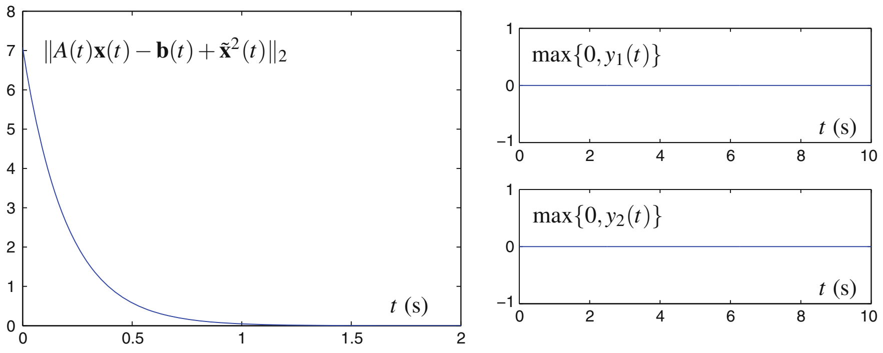


Fig. 6.5 Simulation results via ZD model (6.10) starting from $\mathbf{x}(0) \in \mathbb{S}(0)$ in Example 6.1

For demonstrating the effectiveness of ZD model (6.10) on solving time-varying linear matrix-vector inequality (6.1), the right graph of Fig. 6.5 shows the transient behavior of $\max\{\mathbf{0}, \mathbf{y}(t)\}$. Since the initial state $\mathbf{x}(0)$ is inside the initial solution set $\mathbb{S}(0)$, ZD model (6.10) guarantee that $A(t)\mathbf{x}(t) - \mathbf{b}(t) \leqslant \mathbf{0}$ always holds true without an appreciable convergence process (i.e., $\max\{\mathbf{0}, \mathbf{y}(t)\} = \mathbf{0}$). The right graph of Fig. 6.5 illustrates and substantiates the above situation. The results are consistent with those of the first two ZD models (6.3) and (6.6) (see also [6, 13]).

In summary, from the above simulation results of two cases (i.e., Figs. 6.2 through 6.5), we can conclude that ZD model (6.10) is effective on solving time-varying linear matrix-vector inequality (6.1) and the corresponding time-varying linear matrix-vector equation (6.7), no matter whether the initial state $\mathbf{x}(0)$ is inside or outside the initial solution set $\mathbb{S}(0)$.

Example 6.2 In this example, we exploit ZD model (6.10) with $\gamma = 5$ to solve a more complex time-varying linear inequality (6.1) with time-varying Toeplotz coefficients being as follows:

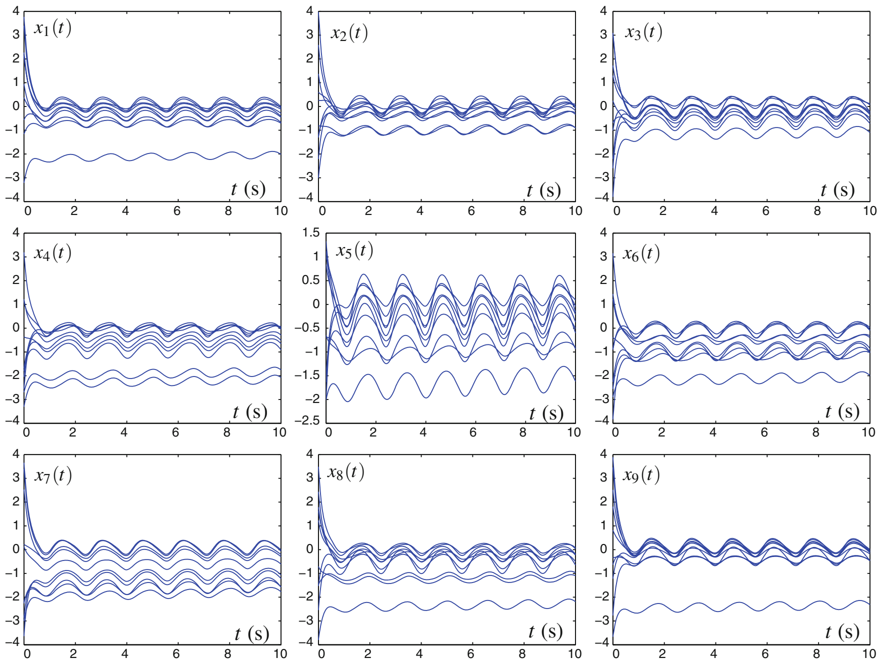


Fig. 6.6 Transient behavior of $\mathbf{x}(t)$ synthesized by ZD model (6.10) starting with 10 randomly-generated initial states $\mathbf{x}(0) \in [-4, 4]^9$ in Example 6.2

$$A(t) = \begin{bmatrix} a_1(t) & a_2(t) & a_3(t) & \dots & a_9(t) \\ a_2(t) & a_1(t) & a_2(t) & \dots & a_8(t) \\ a_3(t) & a_2(t) & a_1(t) & \dots & a_7(t) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_9(t) & a_8(t) & a_7(t) & \dots & a_1(t) \end{bmatrix} \in \mathbb{R}^{9\times 9} \text{ and } \mathbf{b}(t) = \begin{bmatrix} b_1(t) \\ b_2(t) \\ b_3(t) \\ \vdots \\ b_9(t) \end{bmatrix} \in \mathbb{R}^9,$$

where $a_1(t) = 9 + \sin(4t)$ and $a_k(t) = \cos(4t)/(k-1)$ with $k = 2, 3, \dots, 9$. In addition, $b_{2k-1}(t) = \cos(4t) + 2$ with $k = 1, 2, \dots, 5$, and $b_{2k}(t) = \sin(4t) + 1$ with $k = 1, 2, 3, 4$.

Note that, for solving such a complex time-varying linear matrix-vector inequality (6.1), it may be difficult to generate a random initial state such that the $\mathbf{x}(0)$ is inside or outside the initial solution set $\mathbb{S}(0)$ completely. Thus, it is worth investigating the performance of ZD model (6.10) for solving time-varying linear inequality (6.1) when initial states $\mathbf{x}(0)$ are randomly-generated in a more general sense that some elements of $\mathbf{x}(0)$ are inside the initial solution set $\mathbb{S}(0)$, while the others are outside the initial solution set $\mathbb{S}(0)$. By applying ZD model (6.10) with 10 randomly-generated initial states $\mathbf{x}(0)$ to online solution of such a complex time-varying linear matrix-vector inequality (6.1), the corresponding simulation results are illustrated in Figs. 6.6 and 6.7.

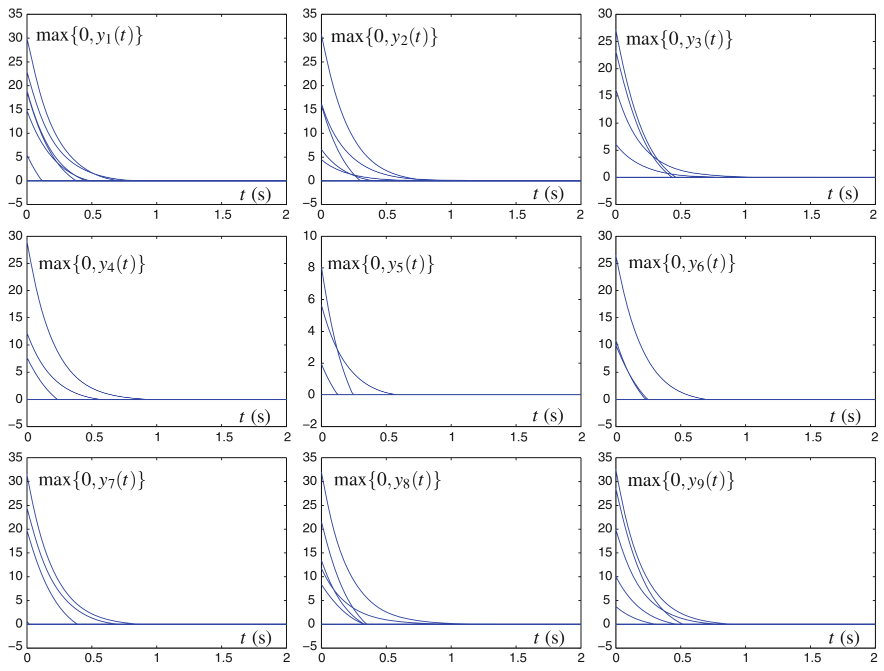


Fig. 6.7 Transient behavior of $\max\{\mathbf{0}, \mathbf{y}(t)\}$ synthesized by ZD model (6.10) starting with 10 randomly-generated initial states $\mathbf{x}(0) \in [-4, 4]^9$ in Example 6.2

As seen from Fig. 6.6, starting from 10 randomly-generated initial states $\mathbf{x}(0) \in [-4, 4]^9$, the solutions synthesized by ZD model (6.10) are all time-varying or possess dynamic performance. In addition, from Fig. 6.7, we can see that $\max\{\mathbf{0}, \mathbf{y}(t)\}$ all converge to zero within 0.5 s. In detail, some $\max\{0, y_i(t)\}$ remain zero corresponding to $y_i(0) \leqslant 0$ (with $i \in \{1, 2, \ldots, 9\}$) and the other $\max\{0, y_j(t)\}$ decrease to zero rapidly corresponding to $y_i(0) > 0$ (with $j \neq i$ and $j \in \{1, 2, \ldots, 9\}$). The results demonstrate again that ZD model (6.10) is effective on solving more complex time-varying linear matrix-vector inequality.

6.4 Summary

In this chapter, focusing on solving time-varying matrix-vector inequality (6.1), we have proposed, generalized, developed, and investigated three different ZD models [i.e., (6.3), (6.6), and (6.10)] by defining three different ZFs [i.e., (6.2), (6.4), and (6.8)]. Theoretical results have also been given to show the excellent convergence performance of the proposed ZD models. Computer simulation results have further substantiated the efficacy of ZD model (6.10) for time-varying linear matrix-vector inequality solving.

References

1. Marashdeh Q, Warsito W, Fan L, Teixeira F (2006) A nonlinear image reconstruction technique for ECT using a combined neural network approach. Meas Sci Technol 17:2097–2103
2. Cichocki A, Bargiela A (1997) Neural network for solving linear inequality systems. Parallel Comput 22:1455–1475
3. Labonte G (1997) On solving systems of linear inequalities with artificial neural network. IEEE Trans Neural Netw 8:590–600
4. Lin C, Lai C, Huang T (2000) A neural network for linear matrix inequality problems. IEEE Trans Neural Netw 11:1078–1092
5. Yang K, Murty KG, Mangasarian OL (1992) New iterative methods for linear inequalities. J Optim Theory Appl 72:163–185
6. Guo D, Zhang Y (2012) A new variant of the Zhang neural network for solving online time-varying linear inequalities. Proc R Soc A 468(2144):2255–2271
7. Zhang Y (2006) A set of nonlinear equations and inequalities arising in robotics and its online solution via a primal neural network. Neurocomputing 70:513–524
8. Zhang Y (2002) Analysis and design of recurrent neural networks and their applications to control and robotic systems. Ph.D. dissertation, Chinese University of Hong Kong, Hong Kong
9. Zhang Y, Wang J (2004) Obstacle avoidance for kinematically redundant manipulators using a dual neural network. IEEE Trans Syst Man Cybern Part B 34:752–759
10. Guo D, Zhang Y (2012) A new inequality-based obstacle-avoidance MVN scheme and its application to redundant robot manipulators. IEEE Trans Syst Man Cybern Part C 42(6):1326–1340
11. Xia Y (1996) A new neural network for solving linear programming problems and its application. IEEE Trans Neural Netw 7:525–529

12. Xia Y, Wang J, Hung DL (1999) Recurrent neural networks for solving linear inequalities and equations. IEEE Trans Circuits Syst I Fundam Theory Appl 46:452–462
13. Xiao L, Zhang Y (2011) Zhang neural network versus gradient neural network for solving time-varying linear inequalities. IEEE Trans Neural Netw 22(10):1676–1684
14. Mathews JH, Fink KD (2004) Numerical methods using MATLAB. Prentice Hall, New Jersey
15. Xiao L, Zhang Y (2013) Different Zhang functions resulting in different ZNN models demonstrated via time-varying linear matrix-vector inequalities solving. Neurocomputing 121:140–149
16. Cichocki A, Bargiela A (1997) Neural network for solving linear inequality systems. Parallel Comput 22:1455–1475
17. Guo D, Zhang Y (2014) Zhang neural network for online solution of time-varying linear matrix inequality aided with an equality conversion. IEEE Trans Neural Netw Learn Syst 25(2):370–382

Part III
Matrix-Valued ZF in Real Domain

Chapter 7
Time-Varying Matrix Inverse

Abstract In this chapter, focusing on time-varying matrix inversion, we propose, generalize, develop, and investigate different ZFs that lead to different ZD models. Meanwhile, a specific relationship between the proposed ZD model and others' model/method [i.e., the Getz and Marsden (G-M) dynamic system] is presented. Eventually, the MATLAB Simulink modeling and simulative verifications with examples using such different ZD models are further researched. Both theoretical analysis and modeling results further substantiate the efficacy of the proposed ZD models which originate from different ZFs for time-varying matrix inversion.

7.1 Introduction

In recent years, the problem of solving linear matrix equations, e.g., Sylvester equation, Lyapunov equation, and Stein's equation, has been encountered in various science and engineering fields [1–5]. As a subtopic of the linear matrix equations solving, matrix inversion is often treated as one of the fundamental issues [5–7], for instance, as preliminary steps for robotic kinematics and optimization.

Generally speaking, if the time-varying matrix $A(t) \in \mathbb{R}^{m\times n}$ is of full-rank, i.e., $\operatorname{rank}(A) = \min\{m, n\}$ at any time instant $t \in [0, +\infty)$, then the unique time-varying pseudoinverse/inverse $A^+(t)$ for matrix $A(t)$ is given as [5, 8–10]

$$A^+(t) = \begin{cases} (A^{\mathrm{T}}(t)A(t))^{-1}A^{\mathrm{T}}(t), & \text{if } m > n, \\ A^{-1}(t), & \text{if } m = n, \\ A^{\mathrm{T}}(t)(A(t)A^{\mathrm{T}}(t))^{-1}, & \text{if } m < n. \end{cases} \tag{7.1}$$

As for (7.1), the upper part, middle part, and lower part correspond to the left pseudoinverse, inverse, and right pseudoinverse, respectively. Note that, in this chapter, we only focus on the investigation of solving for time-varying inverse $A^-(t)$ of a nonsingular square matrix $A(t)$ [i.e., (7.1) under the situation of $m = n$]. The investigations of solving for time-varying left and right pseudoinverse of a full-rank rectangular matrix $A(t)$ [i.e., (7.1) under the situation of $m \neq n$] are conducted, respectively, in the ensuing two Chaps. 8 and 9.

Y. Zhang and D. Guo, *Zhang Functions and Various Models*,
DOI 10.1007/978-3-662-47334-4_7

In mathematics, the problem of matrix inversion is generally formulated as $AX = I \in \mathbb{R}^{n\times n}$. Much effort has been directed toward computational aspects of fast matrix inversion since the mid-1980s, and subsequently lots of numerical algorithms have been proposed for matrix inversion [11–14]. In general, owing to the serial-processing feature performed on digital computers, it may not be efficient enough in large-scale online or real-time applications for most numerical algorithms. Note that, for such numerical algorithms, the minimal arithmetic operations are proportional to the cube of the matrix dimension [15]. In view of this situation, a variety of parallel-processing computational methods (e.g., random neural networks and gradient neural networks) [5–7, 16–20] have been further developed and implemented on specific architectures. It is worth pointing out here that almost all of the reported methods are theoretically/intrinsically designed for time-invariant matrix inversion (instead of time-varying matrix inversion investigated in [5–7]).

Aiming at time-varying matrix inversion, in this chapter, by introducing different ZFs, different ZD models are proposed, generalized, developed, and investigated. Note that a dynamic system has been proposed by Getz and Marsden [21–23], which is described in an explicit dynamics; and that such a dynamic system (termed the G-M dynamic system) can converge exponentially to the theoretical time-varying inverse $A^{-1}(t)$, for sufficiently large design parameter under the condition of hardware permitting and for initial conditions adequately close to the initial theoretical inverse $A^{-1}(0)$. Then, the direct link between the proposed ZD model and the G-M dynamic system is discovered. Apart from the theoretical analysis on the convergence properties of the proposed ZD models, MATLAB Simulink modeling and simulative examples are provided accordingly. Modeling and simulative results further substantiate the efficacy of the proposed ZD models derived from different ZFs for time-varying matrix inversion.

7.2 ZFs and ZD Models

In this section, we introduce different ZFs, propose the resultant ZD models, and show the relationship between the ZD model and the G-M dynamic system for time-varying matrix inversion. Meanwhile, the relevant theoretical analysis is given.

To lay a basis for further discussion, the problem of time-varying matrix inversion investigated in this chapter is described in the following standard form:

$$A(t)X(t) - I = 0 \in \mathbb{R}^{n\times n}, \tag{7.2}$$

where $A(t) \in \mathbb{R}^{n\times n}$ is the smoothly time-varying nonsingular coefficient matrix, $I \in \mathbb{R}^{n\times n}$ is the identity matrix, and $X(t) \in \mathbb{R}^{n\times n}$ is the time-varying unknown matrix to be obtained. The target of this chapter is to find $X(t)$ such that (7.2) holds true for any time instant $t \geqslant 0$, i.e., to invert matrix $A(t)$ in real time $t \geqslant 0$. Note that $A(t)$ together with its time derivative $\dot{A}(t) \in \mathbb{R}^{n\times n}$ is assumed to be known or measurable.

In view of that (7.2) is depicted in the matrix form (which is different from those scalar and matrix vector forms presented in the previous chapters), we denote the corresponding ZF by $E(t)$ with $\dot{E}(t)$ being its time derivative. Note that, in Chaps. 7–10, $E(t)$ and $\dot{E}(t)$ are used as the notations of the matrix-valued ZF and its time derivative, respectively. Thus, the ZD design formula (4.2) presented in Chap. 4 is further generalized as follows (i.e., from the vector form to the matrix form) [5–7]:

$$\dot{E}(t) = \frac{\mathrm{d}E(t)}{\mathrm{d}t} = -\gamma E(t), \tag{7.3}$$

where design parameter $\gamma \in \mathbb{R}$ is defined the same as before. Based on different ZFs and the ZD design formula (7.3), the resultant ZD models are developed and investigated for time-varying matrix inversion [i.e., (7.2)].

Specifically, for solving time-varying matrix-inversion problem (7.2), in this chapter, we define different ZFs as below:

$$E(t) = A^{-1}(t) - X(t), \tag{7.4}$$
$$E(t) = A(t) - X^{-1}(t), \tag{7.5}$$
$$E(t) = A(t)X(t) - I, \tag{7.6}$$
$$E(t) = X(t)A(t) - I, \tag{7.7}$$
$$E(t) = (A(t)X(t))^{-1} - I, \tag{7.8}$$
$$E(t) = (X(t)A(t))^{-1} - I. \tag{7.9}$$

Before constructing different ZD models from different ZFs, we present the following theorem for further discussion.

Theorem 7.1 *We have the following facts:*

$$\frac{\mathrm{d}(X^{-1}(t))}{\mathrm{d}t} = -X^{-1}(t)\dot{X}(t)X^{-1}(t), \tag{7.10}$$

$$\frac{\mathrm{d}(A^{-1}(t))}{\mathrm{d}t} = -A^{-1}(t)\dot{A}(t)A^{-1}(t), \tag{7.11}$$

$$\frac{\mathrm{d}(A(t)X(t))^{-1}}{\mathrm{d}t} = -(A(t)X(t))^{-1}\frac{\mathrm{d}(A(t)X(t))}{\mathrm{d}t}(A(t)X(t))^{-1}. \tag{7.12}$$

Proof It is obtained readily by following the proof of Theorem 4.1 in Chap. 4. □

According to the ZD model design formula (7.3), with six different ZFs [i.e., (7.4)–(7.9)] used, six different ZD models for time-varying matrix inversion are thus derived and presented as follows.

- Considering ZD design formula (7.3), ZF (7.4), and Eq. (7.11), we then have

$$A(t)\dot{X}(t)A(t) = -\gamma(A(t)X(t) - I)A(t) - \dot{A}(t), \tag{7.13}$$

  which is also rewritten in the following explicit form:

$$\dot{X}(t) = \dot{X}(t) + (A(t)\dot{X}(t) - \gamma(A(t)X(t) - I))A(t) + \dot{A}(t).$$

  Therefore, based on ZF (7.4), we obtain ZD model (7.13) for time-varying matrix inversion.
- Based on ZD design formula (7.3), ZF (7.5), and Eq. (7.10) we have

$$\dot{X}(t) = -X(t)\dot{A}(t)X(t) - \gamma X(t)(A(t)X(t) - I). \tag{7.14}$$

  Therefore, we obtain ZD model (7.14) based on ZF (7.5), which is exactly the G-M dynamic system for time-varying matrix inversion [21–23]. In other words, a direct link between the ZD model and others' model/method (i.e., the G-M dynamic system) for time-varying matrix inversion is found. Specifically, the G-M dynamic system can be derived directly from the ZD design formula (7.3) with ZF (7.5) exploited. Thus, the G-M dynamic system can be regarded as a special case of ZD models. Evidently, this chapter shows an explanation to the G-M dynamic system for time-varying matrix inversion, which is different from the original derivation proposed by Getz and Marsden [21–23].
- With ZD design formula (7.3) and ZF (7.6) exploited, the following ZD model is established:

$$A(t)\dot{X}(t) = -\dot{A}(t)X(t) - \gamma(A(t)X(t) - I), \tag{7.15}$$

  and similarly we have the following explicit form:

$$\dot{X}(t) = (I - A(t))\dot{X}(t) - \dot{A}(t)X(t) - \gamma(A(t)X(t) - I).$$

- Based on ZD design formula (7.3) and ZF (7.7), the resultant ZD model is expressed as below:

$$\dot{X}(t)A(t) = -X(t)\dot{A}(t) - \gamma(X(t)A(t) - I), \tag{7.16}$$

  of which the explicit form is shown as follows:

$$\dot{X}(t) = \dot{X}(t)(I - A(t)) - X(t)\dot{A}(t) - \gamma(X(t)A(t) - I).$$

- Considering ZD design formula (7.3), ZF (7.8), and Eq. (7.12), we have

$$A(t)\dot{X}(t) = -\dot{A}(t)X(t) - \gamma(A(t)X(t) - I)A(t)X(t), \tag{7.17}$$

Table 7.1 Different ZFs resulting in different ZD models (depicted in explicit dynamics for modeling purposes) for time-varying matrix inversion

ZF	ZD model
(7.4)	$\dot{X}(t)=\dot{X}(t)+(A(t)\dot{X}(t)+\gamma(A(t)X(t)-I))A(t)+\dot{A}(t)$
(7.5)	$\dot{X}(t)=-X(t)\dot{A}(t)X(t)-\gamma X(t)(A(t)X(t)-I)$
(7.6)	$\dot{X}(t)=(I-A(t))\dot{X}(t)-\dot{A}(t)X(t)-\gamma(A(t)X(t)-I)$
(7.7)	$\dot{X}(t)=\dot{X}(t)(I-A(t))-X(t)\dot{A}(t)-\gamma(X(t)A(t)-I)$
(7.8)	$\dot{X}(t)=(I-A(t))\dot{X}(t)-\dot{A}(t)X(t)-\gamma(A(t)X(t)-I)A(t)X(t)$
(7.9)	$\dot{X}(t)=\dot{X}(t)(I-A(t))-X(t)\dot{A}(t)-\gamma X(t)A(t)(X(t)A(t)-I)$

and then we further have the following explicit form:

$$\dot{X}(t)=(I-A(t))\dot{X}(t)-\dot{A}(t)X(t)-\gamma(A(t)X(t)-I)A(t)X(t).$$

- Similarly, based on ZF (7.9), we have

$$\dot{X}(t)A(t)=-X(t)\dot{A}(t)-\gamma X(t)A(t)(X(t)A(t)-I), \qquad (7.18)$$

of which the explicit form is shown below:

$$\dot{X}(t)=\dot{X}(t)(I-A(t))-X(t)\dot{A}(t)-\gamma X(t)A(t)(X(t)A(t)-I).$$

Thus, we obtain six different types of ZD models [i.e., ZD models (7.13)–(7.18)] for time-varying matrix inversion [with the problem formulated as (7.2)], which are based on six different types of ZFs [i.e., ZFs (7.4)–(7.9)]. For readers' convenience and also for comparison, such six different ZD models corresponding to six different ZFs are listed in Table 7.1.

7.3 Theoretical Results and Analyses

In this section, from the theoretical results of [20, 23], we summarize and present the following general observations on the convergence properties of ZD models (7.13)–(7.18) for time-varying matrix inversion [i.e., (7.2)].

Theorem 7.2 *Let us consider a smoothly time-varying nonsingular matrix $A(t)\in\mathbb{R}^{n\times n}$ in (7.2). Starting from an initial state $X(0)\in\mathbb{R}^{n\times n}$, the state matrix $X(t)$ of ZD model (7.13) derived from ZF (7.4) globally and exponentially converges to the theoretical time-varying inverse $A^{-1}(t)$ of matrix $A(t)$.*

Proof From the compact form of the presented ZD design formula $\dot{E}(t) = -\gamma E(t)$, a set of $n \times n$ decoupled differential equations is written equivalently as follows:

$$\dot{e}_{ij}(t) = -\gamma e_{ij}(t), \tag{7.19}$$

for any $i, j \in \{1, 2, 3, \ldots, n\}$. Thus, we define a Lyapunov function candidate $v_{ij}(t) = e_{ij}^2(t)/2 \geqslant 0$ with its time derivative being

$$\dot{v}_{ij}(t) = \frac{\mathrm{d}v_{ij}(t)}{\mathrm{d}t} = e_{ij}(t)\dot{e}_{ij}(t) = -\gamma e_{ij}^2(t) \leqslant 0,$$

which guarantees the negative-definiteness of $\dot{v}_{ij}$ (i.e., $\dot{v}_{ij} < 0$ for $e_{ij} \neq 0$ while $\dot{v}_{ij} = 0$ for $e_{ij} = 0$ only). By Lyapunov theory, equilibrium point $e_{ij} = 0$ of (7.19) is globally asymptotically stable; i.e., $e_{ij}(t)$ globally converges to zero, for any $i, j \in \{1, 2, 3, \ldots, n\}$. In other words, the matrix-valued error function $E(t) = [e_{ij}(t)] \in \mathbb{R}^{n\times n}$ is globally convergent to zero. In addition, we have $E(t) = A^{-1}(t) - X(t)$; or equivalently, $X(t) = A^{-1}(t) - E(t)$. Since $E(t) \to 0$ as $t \to +\infty$, we have $X(t) \to A^{-1}(t)$ as $t \to +\infty$. That is, state matrix $X(t)$ of ZD model (7.13) derived from ZF (7.4) globally converges to the theoretical time-varying inverse $A^{-1}(t)$ of matrix $A(t)$. The proof on global convergence is thus complete.

Furthermore, in view of $\dot{e}_{ij} = -\gamma e_{ij}$, solving the linear first-order differential equation yields readily $e_{ij}(t) = \exp(-\gamma t)e_{ij}(0)$. In other words, the matrix-valued error function $E(t) \in \mathbb{R}^{n\times n}$ is expressed explicitly as

$$E(t) = \begin{bmatrix} e_{11}(0) & e_{12}(0) & \cdots & e_{1n}(0) \\ e_{21}(0) & e_{22}(0) & \cdots & e_{2n}(0) \\ \vdots & \vdots & \ddots & \vdots \\ e_{n1}(0) & e_{n2}(0) & \cdots & e_{nn}(0) \end{bmatrix} \exp(-\gamma t) = E(0)\exp(-\gamma t).$$

With $\alpha = E(0)$ and in view of ZF (7.4), the above equation is further rewritten as

$$A^{-1}(t) - X(t) = \alpha \exp(-\gamma t),$$

which indicates that $X(t)$ exponentially converges to $A^{-1}(t)$ with convergence rate $\gamma > 0$. That is, starting from an initial state $X(0) \in \mathbb{R}^{n\times n}$, the state matrix $X(t)$ of ZD model (7.13) derived from ZF (7.4) exponentially converges to the theoretical time-varying matrix inverse $A^{-1}(t)$.

In summary, the state matrix $X(t)$ of ZD model (7.13), starting from an initial state $X(0)$, globally and exponentially converges to the the theoretical time-varying inverse $A^{-1}(t)$ of matrix $A(t)$. The proof is thus complete. $\square$

As for the other five ZD models (7.14)–(7.18), we also have the following convergence results, with the related proofs being generalized from the proof of Theorem 7.2 and being left to interested readers to complete as a topic of exercise.

Proposition 7.1 *Consider a smoothly time-varying nonsingular matrix $A(t) \in \mathbb{R}^{n\times n}$ in (7.2). Starting from an initial state $X(0) \in \mathbb{R}^{n\times n}$ that is close enough to the initial theoretical inverse $A^{-1}(0)$, the state matrix $X(t)$ of ZD model (7.14) derived from ZF (7.5), the state matrix $X(t)$ of ZD model (7.17) derived from ZF (7.8), and the state matrix $X(t)$ of ZD model (7.18) derived from ZF (7.9) exponentially converge to the theoretical time-varying inverse $A^{-1}(t)$ of matrix $A(t)$.*

Proposition 7.2 *Consider a smoothly time-varying nonsingular matrix $A(t) \in \mathbb{R}^{n\times n}$ in (7.2). Starting from an initial state $X(0) \in \mathbb{R}^{n\times n}$, the state matrix $X(t)$ of ZD model (7.15) derived from ZF (7.6) and the state matrix $X(t)$ of ZD model (7.16) derived from ZF (7.7) globally and exponentially converge to the theoretical time-varying inverse $A^{-1}(t)$ of matrix $A(t)$.*

7.4 Simulink Modeling

For possible circuit implementation and also for the final purpose of FPGA and ASIC realization, the MATLAB Simulink modeling of the proposed ZD models (7.13)–(7.18) is researched in this section.

Specifically, the corresponding block diagrams of such ZD models are shown in Figs. 7.1, 7.2, and 7.3. It is worth noting that, in order to make clear the block diagrams of ZD models (7.15) and (7.16) shown in Fig. 7.2 as well as ZD models (7.17) and (7.18) shown in Fig. 7.3, we indicate the left multiplication and the right multiplication via the position of the symbol "$*$", e.g., "$A(t)*$" and "$*A(t)$" stand for "$A(t) * X(t)$" and "$X(t) * A(t)$", respectively. In addition, we take ZD model (7.15)

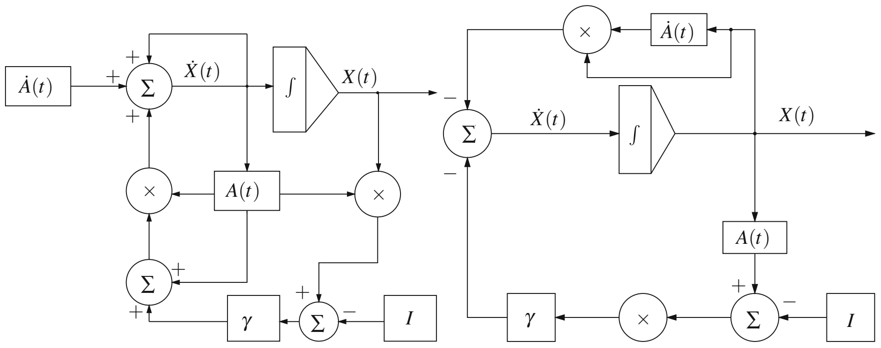


Fig. 7.1 Block diagrams of ZD models (7.13) and (7.14) for time-varying matrix inversion

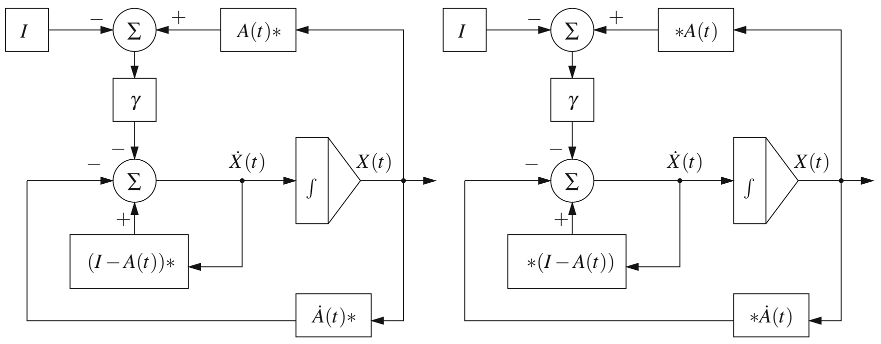


Fig. 7.2 Block diagrams of ZD models (7.15) and (7.16) for time-varying matrix inversion

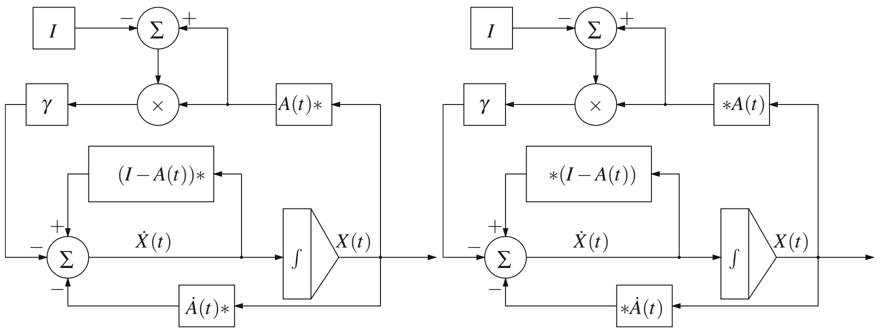


Fig. 7.3 Block diagrams of ZD models (7.17) and (7.18) for time-varying matrix inversion

as an example for further investigation on hardware implementation. Evidently, it follows from (7.15) that the *ij*th neuron dynamics of ZD model (7.15) is expressed as the following dynamic equation:

$$\dot{x}_{ij} = \sum_{k=1}^{n} \mu_{ik}\dot{x}_{kj} - \sum_{k=1}^{n} \dot{a}_{ik}x_{kj} - \gamma\left(\sum_{k=1}^{n}(a_{ik}x_{kj} - \delta_{ij})\right),$$

where

- x_{ij} denotes the *ij*th neuron state of ZD model (7.15) corresponding to the *ij*th entry of state matrix $X(t)$, with $i, j = 1, 2, \dots, n$;
- time-variant weights a_{ij} and $\dot{a}_{ij}$ are defined, respectively, as the *ij*th entries of matrix $A(t)$ and its time-derivative measurement $\dot{A}(t)$;
- δ_{ij} is the Kronecker delta defined here as the *ij*th entry of the identity matrix I, and $\mu_{ij} = \delta_{ij} - a_{ij}$.

The jth-column circuit schematic of ZD model (7.15) is thus shown in Fig. 7.4.

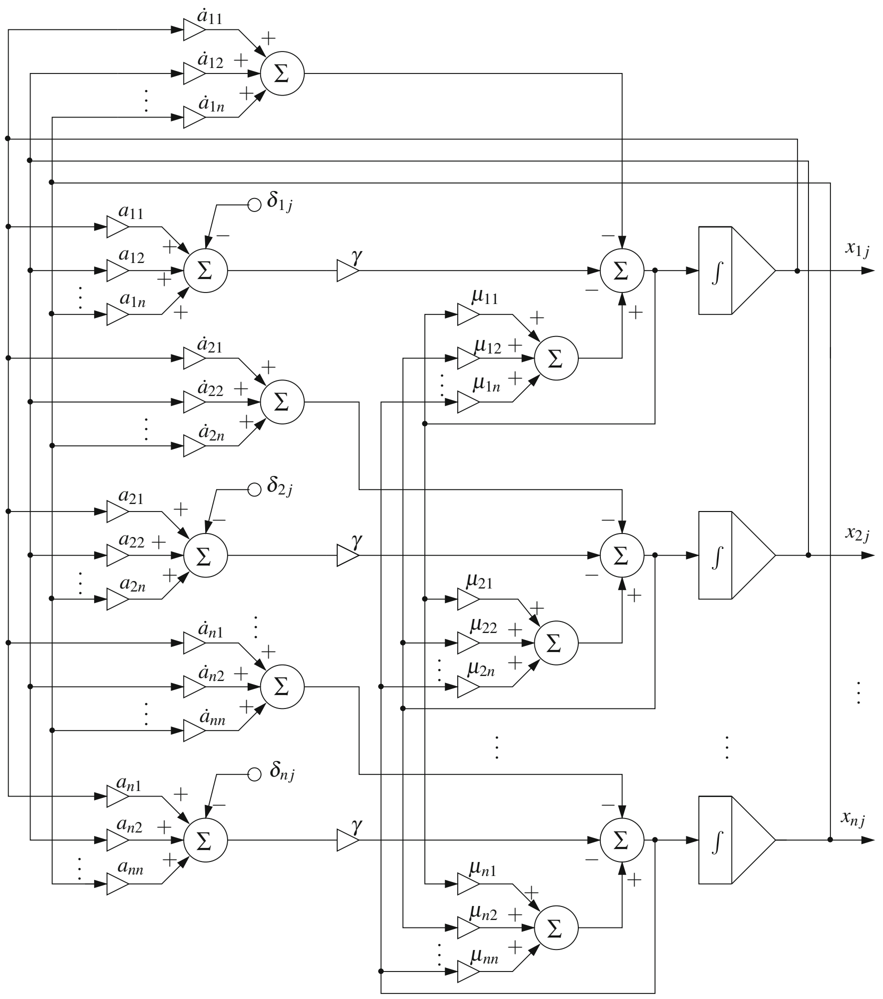


Fig. 7.4 Circuit schematic of the jth column (with $j = 1, 2, \ldots, n$) of neurons of ZD model (7.15)

Hence, based on the above analysis, the overall Simulink modeling of ZD model (7.15) for time-varying matrix inversion is shown in Fig. 7.5, where $A(t)$ is generated by employing the "MATLAB Function" block with the "Clock" block used as its input. Note that other Simulink modeling of ZD models [i.e., (7.13) and (7.14), and (7.16)–(7.18)] can also be obtained through the above processing, but omitted here and left to interested readers to complete as a topic of exercise.

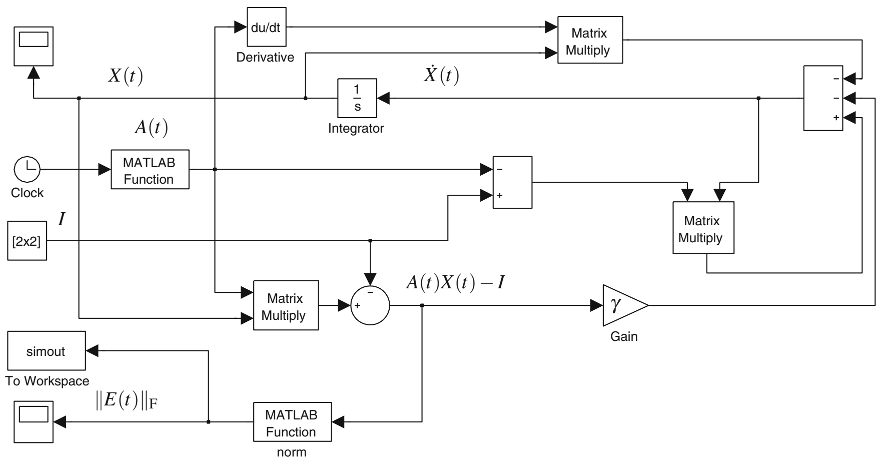


Fig. 7.5 Overall Simulink modeling of ZD model (7.15) for time-varying matrix inversion

7.5 Illustrative Examples

In the previous sections, six different ZD models (7.13)–(7.18) based on different ZFs are presented, studied, and modeled for time-varying matrix inversion, together with corresponding theoretical analysis and results. In this section, based on the above-mentioned overall Simulink modeling technique, the following illustrative examples are provided to show the efficacy of the proposed ZD models for time-varying matrix inversion.

Example 7.1 Let us consider the time-varying matrix-inversion problem (7.2) [i.e., $A(t)X(t) = I$] with the following time-varying matrix $A(t)$:

$$A(t) = \begin{bmatrix} \sin(5t) & \cos(5t) \\ -\cos(5t) & \sin(5t) \end{bmatrix} \in \mathbb{R}^{2\times 2}. \tag{7.20}$$

By algebraic operations, the theoretical time-varying inverse of $A(t)$ is given as

$$X^*(t) = A^{-1}(t) = \begin{bmatrix} \sin(5t) & -\cos(5t) \\ \cos(5t) & \sin(5t) \end{bmatrix} \in \mathbb{R}^{2\times 2}.$$

Thus, we can use such a theoretical solution to compare with the solutions of corresponding ZD models and then check the correctness of the models' solutions.

The proposed ZD models (7.13)–(7.18) are exploited to solve such a problem, and the corresponding simulation results based on ZD model (7.13) are illustrated in Fig. 7.6. Specifically, as shown in the left graph of Fig. 7.6, with design parameter

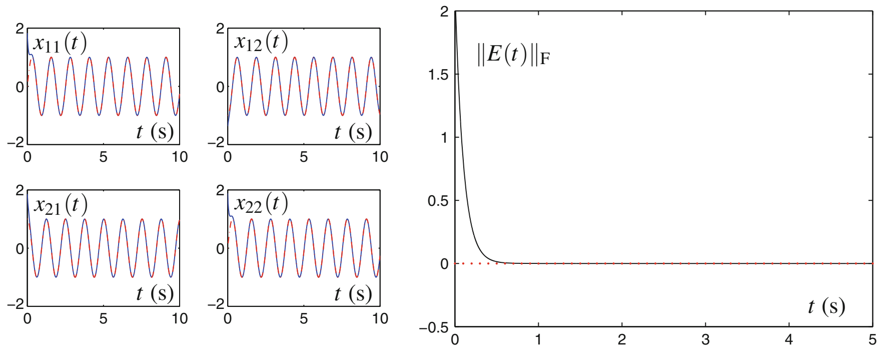


Fig. 7.6 Convergence performance of ZD model (7.13) with $\gamma = 10$ for inverting the time-varying matrix $A(t)$ in (7.20)

$\gamma = 10$, state matrix $X(t)$ of ZD model (7.13) denoted by solid curves converges rapidly to the theoretical solution $X^*(t)$ denoted by dash-dotted curves. In addition, to further investigate the convergence performance of ZD model (7.13), we monitor the residual error $\|E(t)\|_F = \|A(t)X(t) - I\|_F$ (with $\|\cdot\|_F$ denoting the Frobenius norm of a matrix) during the inverting process. As seen from the right graph of Fig. 7.6, by applying (7.13) to inverting the time-varying matrix (7.20), the residual error converges to zero fast and accurately in about 1 s. Note that, as for other ZD models [i.e., (7.14)–(7.18)], we have the same or similar observations which are omitted because of results similarity. The modeling testings of such ZD models are left to interested readers to complete as a topic of exercise.

Example 7.2 To further substantiate the efficacy of the proposed ZD models (7.13)–(7.18) for more complicated situations, we consider (7.2) with the following time-varying Toeplitz matrix $A(t)$:

$$A(t) = \begin{bmatrix} a_1(t) & a_2(t) & \cdots & a_n(t) \\ a_2(t) & a_1(t) & \cdots & a_{n-1}(t) \\ \vdots & \vdots & \ddots & \vdots \\ a_n(t) & a_{n-1}(t) & \cdots & a_1(t) \end{bmatrix} \in \mathbb{R}^{n\times n}. \tag{7.21}$$

Let $a_1(t) = n + \sin(5t)$ and $a_k(t) = \cos(5t)/(k-1)$ with $k = 2, 3, \ldots, n$. Evidently, Toeplitz matrix $A(t)$ is strictly diagonally dominant for any time instant $t \geqslant 0$ and is thus invertible. Figure 7.7 shows the simulation results by using ZD models (7.13)–(7.18) with $\gamma = 10$ for the time-varying inverse of the above Toeplitz matrix $A(t)$ under the condition of $n = 4$. As seen from the figure, residual errors $\|E(t)\|_F$ of the proposed ZD models all converge to zero, which means that their corresponding state matrices converge to $A^{-1}(t)$. These results manifest again the efficacy of the proposed ZD models (7.13)–(7.18) on time-varying matrix inversion.

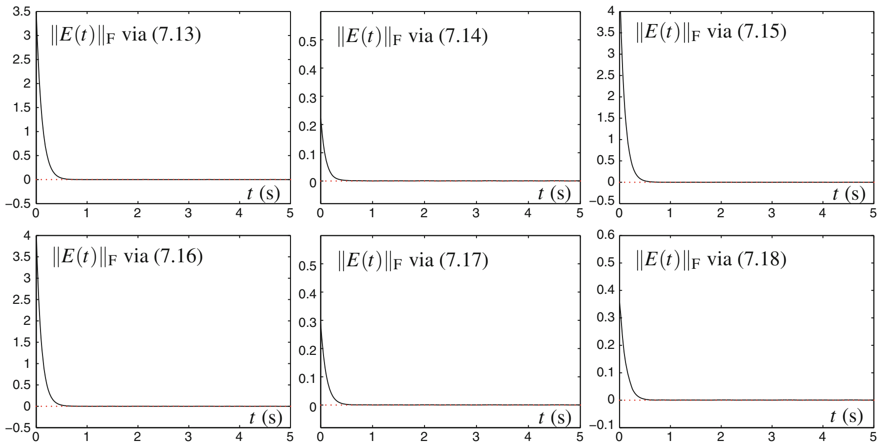


Fig. 7.7 Residual errors $\|E(t)\|_F$ of ZD models (7.13)–(7.18) for inverting the time-varying Toeplitz matrix $A(t)$ in (7.21)

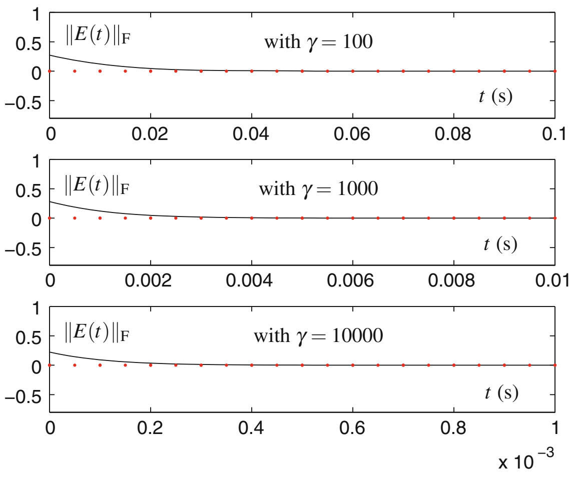


Fig. 7.8 Residual errors $\|E(t)\|_F$ of ZD model (7.17) using different values of γ (i.e., $\gamma = 100$, 1000, and 10000) for inverting the time-varying Toeplitz matrix $A(t)$ in (7.21)

Meanwhile, the residual errors $\|E(t)\|_F$ of ZD model (7.17) with different values of γ are illustrated in Fig. 7.8. As shown in the figure, the convergence time of ZD model (7.17) can be expedited from around 0.04 to 0.004 and even to 0.0004 s, when the γ value is increased from 100 to 1000 and to 10,000, respectively. This observation tells that ZD model (7.17) has an exponential convergence property, which can be expedited effectively by increasing the value of γ. Note that, for other ZD models [i.e., (7.13)–(7.16) and (7.18)], we have the same or similar observations which are omitted here due to results similarity. Besides, the modeling testings of ZD models (7.13)–(7.16) and (7.18) using different γ values are left to interested readers to complete as a topic of exercise.

In summary, the above simulation results have shown the efficacy of the proposed ZD models (7.13)–(7.18) which are derived from different ZFs for time-varying matrix inversion [with the problem formulated as (7.2)]. In addition, they have substantiated the main points of the theoretical analyses which are presented in Sect. 7.3.

7.6 Summary

In this chapter, originating from different ZFs [i.e., (7.4)–(7.9)] as error basis functions, different ZD models (7.13)–(7.18) have been derived, analyzed, and simulated to solve for time-varying matrix inverse. In addition, the clear and direct link between the ZD model and the G-M dynamic system has been found and presented. Moreover, theoretical analysis and results have been given to substantiate the exponential convergence properties of the proposed ZD models for time-varying matrix inversion. Besides, for possible hardware implementation based on electronic circuits, the MATLAB Simulink modeling of the proposed ZD models has been shown and studied. Through computer simulations and illustrative examples, the efficacy of the proposed ZD models (7.13)–(7.18) has been further substantiated for time-varying matrix inversion [i.e., (7.2)].

References

1. Gu C, Xue H (2009) A shift-splitting hierarchical identification method for solving Lyapunov matrix equations. Linear Algebra Appl 430(5–6):1517–1530
2. Zhang Y, Jiang D, Wang J (2002) A recurrent neural network for solving Sylvester equation with time-varying coefficients. IEEE Trans Neural Netw 13(5):1053–1063
3. Zhou B, Lam J, Duan GR (2009) On Smith-type iterative algorithms for the Stein matrix equation. Appl Math Lett 22(7):1038–1044
4. Fuhrmann PA (2010) A functional approach to the Stein equation. Linear Algebra Appl 432(12):3031–3071
5. Zhang Y, Yi C (2011) Zhang neural networks and neural-dynamic method. Nova Science Publishers, New York
6. Guo D, Zhang Y (2012) Zhang neural network, Getz-Marsden dynamic system, and discrete-time algorithms for time-varying matrix inversion with application to robots' kinematic control. Neurocomputing 97:22–32
7. Guo D, Qiu B, Ke Z, Yang Z, Zhang Y (2014) Case study of Zhang matrix inverse for different ZFs leading to different nets. In: Proceedings of the international joint conference on neural networks, pp 2764–2769
8. Zhang Y, Guo D, Li Z (2013) Common nature of learning between back-propagation and Hopfield-type neural networks for generalized matrix inversion with simplified models. IEEE Trans Neural Netw Learn Syst 24(4):579–592
9. Ben-Israel A, Greville TNE (2003) Generalized inverses: theory and applications, 2nd edn. Springer, New York
10. Zhang Y, Yang Y, Tan N, Cai B (2011) Zhang neural network solving for time-varying full-rank matrix Moore-Penrose inverse. Computing 92(2):97–121

11. Yeung KS, Kumbi F (1988) Symbolic matrix inversion with application to electronic circuits. IEEE Trans Circuits Syst 35(2):235–238
12. El-Amawy A (1989) A systolic architecture for fast dense matrix inversion. IEEE Trans Comput 38(3):449–455
13. Wang YQ, Gooi HB (1997) New ordering methods for space matrix inversion via diagonalization. IEEE Trans Power Syst 12(3):1298–1305
14. Mathews JH, Fink KD (2004) Numerical methods using MATLAB. Prentice Hall, New Jersey
15. Koc CK, Chen G (1994) Inversion of all principal submatrices of a matrix. IEEE Trans Aerosp Electr Syst 30(1):280–281
16. Gelenbe E, Hussain KF (2002) Learning in the multiple class random neural network. IEEE Trans Neural Netw 13(6):1257–1267
17. Gelenbe E, Timotheou S (2008) Random neural networks with synchronized interactions. Neural Comput 20(9):2308–2324
18. Wang J (1993) A recurrent neural network for real-time matrix inversion. Appl Math Comput 55(1):89–100
19. Zhang Y, Shi Y, Chen K, Wang C (2009) Global exponential convergence and stability of gradient-based neural network for online matrix inversion. Appl Math Comput 215(3):1301–1306
20. Zhang Y, Ma W, Cai B (2009) From Zhang neural network to Newton iteration for matrix inversion. IEEE Trans Circuits Syst I: Regul Pap 56(7):1405–1415
21. Getz NH, Marsden JE (1995) A dynamic inverse for nonlinear maps. In: Proceedings of 34th IEEE conference on decision and control, pp 4218–4223
22. Getz NH, Marsden JE (1995) Joint-space tracking of workspace trajectories in continuous time. In: Proceedings of 34th IEEE conference on decision and control, pp 1001–1006
23. Getz NH, Marsden JE (1997) Dynamical methods for polar decomposition and inversion of matrices. Linear Algebra Appl 258:311–343

Chapter 8
Time-Varying Matrix Left Pseudoinverse

Abstract In this chapter, focusing on time-varying matrix left pseudoinversion, we propose, generalize, develop, and investigate five different ZD models by introducing five different ZFs. In addition, the link between the ZD models and the Getz–Marsden (G-M) dynamic system is discovered and presented for time-varying matrix left pseudoinversion. Computer simulation results further substantiate the theoretical analysis and show the effectiveness of the proposed ZD models derived from different ZFs on solving for the time-varying matrix left pseudoinverse.

8.1 Introduction

The solution of pseudoinverse (also known as Moore–Penrose generalized inverse) is one of the basic problems encountered in a variety of science and engineering fields, e.g., image noise reduction [1], signal processing [2], robotics [3], linear classifiers [4], and associative memories [5]. Owing to its important roles, many related numerical algorithms have been put forward by researchers [6–9]. For example, Perković and Stanimirović developed an iterative algorithm for estimating the Moore–Penrose generalized inverse [6]. Courrieu proposed an algorithm based on a full-rank Cholesky factorization for fast computation of Moore–Penrose inverse matrices [7]. However, these serial-processing algorithms may be less favorable in large-scale online or real-time applications. Particularly, when applied to the online solution of the time-varying matrix pseudoinverse [10–13], these related iterative algorithms should be performed within every sampling period and the algorithms fail when the sampling rate is too high to allow the algorithms to complete the calculation in a single sampling period.

To lay a basis for further discussion, some necessary preliminaries of the time-varying matrix pseudoinverse are given [13–15].

Y. Zhang and D. Guo, *Zhang Functions and Various Models*,
DOI 10.1007/978-3-662-47334-4_8

Definition 8.1 For a given time-varying matrix $A(t) \in \mathbb{R}^{m\times n}$ with $m \neq n$, if $X(t) \in \mathbb{R}^{n\times m}$ satisfies all of the following four Penrose equations:

$$\begin{aligned} &A(t)X(t)A(t) = A(t), \quad X(t)A(t)X(t) = X(t),\\ &(A(t)X(t))^{\mathrm{T}} = A(t)X(t), \quad (X(t)A(t))^{\mathrm{T}} = X(t)A(t), \end{aligned}$$

then $X(t)$ is called the time-varying pseudoinverse of $A(t)$, which is often denoted by $A^{+}(t)$.

Note that the time-varying pseudoinverse $A^{+}(t)$ always exists and is unique. Specially, if matrix $A(t)$ is of full-rank at any time instant t, i.e., $\text{rank}(A(t)) = \min\{m, n\}$ with $t \in [0, \infty)$, we have the following theorem to obtain the time-varying pseudoinverse of matrix $A(t)$ [13–15].

Theorem 8.1 *For a given time-varying matrix $A(t) \in \mathbb{R}^{m\times n}$ with $m \neq n$, if it satisfies that rank$(A(t)) = \min\{m, n\}$ at any time instant t, then the unique time-varying pseudoinverse $A^{+}(t)$ is given as follows:*

$$A^{+}(t) = \begin{cases} (A^T(t)A(t))^{-1}A^T(t), & \textit{if } m > n,\\ A^T(t)(A(t)A^T(t))^{-1}, & \textit{if } m < n. \end{cases} \tag{8.1}$$

Besides, as for the unique time-varying pseudoinverse of a full-rank matrix, we have another important theorem as follows (which motivates us to define many more ZFs for time-varying matrix pseudoinversion).

Theorem 8.2 *For a given time-varying matrix $A(t) \in \mathbb{R}^{m\times n}$ with $m \neq n$, if it satisfies that rank$(A(t)) = \min\{m, n\}$ at any time instant t, then the unique time-varying pseudoinverse $A^{+}(t)$ is also given as follows:*

$$A^{+}(t) = \begin{cases} \lim\limits_{\mu\to 0} A^T(t)(A(t)A^T(t) + \mu I)^{-1}, & \textit{if } m > n\\ \lim\limits_{\mu\to 0} (A^T(t)A(t) + \mu I)^{-1}A^T(t), & \textit{if } m < n, \end{cases} \tag{8.2}$$

where $\mu > 0 \in \mathbb{R}$.

Proof It can be generalized from [13–15]. □

Evidently, from the above two theorems, we know that there are two cases of time-varying pseudoinverse $A^{+}(t) \in \mathbb{R}^{n\times m}$ for matrix $A(t) \in \mathbb{R}^{m\times n}$ with $m \neq n$. Specifically, $A^{+}(t)$ is termed as the time-varying matrix left pseudoinverse for $A(t)$ with $m > n$; and $A^{+}(t)$ is termed as the time-varying matrix right pseudoinverse for $A(t)$ with $m < n$. Note that, in this chapter, we only consider the smoothly time-varying full-rank matrix $A(t) \in \mathbb{R}^{m\times n}$ with $m > n$, which, together with its time derivative, is assumed to be known or can be estimated accurately. That is to say, this chapter focuses on solving for time-varying matrix left pseudoinverse of

$A(t)$. In the case of $m < n$, the procedure of obtaining the time-varying matrix right pseudoinverse of $A(t)$ is presented in Chap. 9.

More specifically, by introducing five different ZFs, this chapter proposes, generalizes, develops, and investigates five different ZD models for time-varying matrix left pseudoinversion. In addition, the link between the ZD models and the G-M dynamic system is discovered and presented to solve for time-varying matrix left pseudoinverse. Computer simulation results further substantiate the theoretical analysis and show the effectiveness of the proposed ZD models derived from different ZFs for time-varying matrix left pseudoinversion.

8.2 ZFs and ZD Models

In this section, we introduce five different ZFs and propose the resultant ZD models for time-varying matrix left pseudoinversion (together with the theoretical results and the link to the G-M dynamic system).

By denoting $X(t) \in \mathbb{R}^{n\times m}$ as the unknown matrix to be obtained, this chapter aims at designing different ZFs to construct various ZD models to solve for the time-varying matrix left pseudoinverse. That is to say, the unknown $X(t)$ can be obtained by using ZD models in real time t such that it can converge to the exact theoretical time-varying left pseudoinverse $A^+(t)$ (which satisfies all of the above-presented Penrose equations at any time instant t).

8.2.1 *The First ZF and ZD Model*

In order to solve for the time-varying matrix left pseudoinverse, the first ZF (i.e., a matrix-valued unbounded error function) is defined as follows:

$$E(t) = A^{\mathrm{T}}(t)A(t)X(t) - A^{\mathrm{T}}(t) \in \mathbb{R}^{n\times m}. \tag{8.3}$$

Based on ZF (8.3), adopting the ZD design formula (7.3) and in view of $\dot{E}(t) = A^{\mathrm{T}}(t)A(t)\dot{X}(t) + (\dot{A}^{\mathrm{T}}(t)A(t) + A^{\mathrm{T}}(t)\dot{A}(t))X(t) - \dot{A}^{\mathrm{T}}(t)$, we obtain the following ZD model for time-varying matrix left pseudoinversion:

$$\begin{aligned}&A^{\mathrm{T}}(t)A(t)\dot{X}(t)\\ &\quad= \dot{A}^{\mathrm{T}}(t) - (\dot{A}^{\mathrm{T}}(t)A(t) + A^{\mathrm{T}}(t)\dot{A}(t))X(t) - \gamma(A^{\mathrm{T}}(t)A(t)X(t) - A^{\mathrm{T}}(t)),\end{aligned} \tag{8.4}$$

where $X(t) \in \mathbb{R}^{n\times m}$, starting from an initial state $X(0) \in \mathbb{R}^{n\times m}$, denotes the state matrix corresponding to the theoretical time-varying left pseudoinverse $A^+(t)$.

As for ZD model (8.4), we have the following theorem about its convergence property on time-varying matrix left pseudoinversion.

Theorem 8.3 *Given a smoothly time-varying full-rank matrix $A(t) \in \mathbb{R}^{m\times n}$ with $m > n$, the state matrix $X(t)$ of ZD model (8.4), starting from any initial state $X(0)$, converges to the theoretical time-varying left pseudoinverse $A^+(t)$ of matrix $A(t)$.*

Proof Let us define $\tilde{X}(t) = X(t) - A^+(t) \in \mathbb{R}^{n\times m}$ which denotes the difference between the solution $X(t)$ generated by ZD model (8.4) and the theoretical pseudoinverse $A^+(t)$. Following equation $A^{\mathrm{T}}(t)A(t)A^+(t) - A^{\mathrm{T}}(t) = 0 \in \mathbb{R}^{n\times m}$, we obtain its time derivative

$$A^{\mathrm{T}}(t)A(t)\dot{A}^+(t) + (\dot{A}^{\mathrm{T}}(t)A(t) + A^{\mathrm{T}}(t)\dot{A}(t))A^+(t) - \dot{A}^{\mathrm{T}}(t) = 0$$

Substituting $A^+(t) = X(t) - \tilde{X}(t)$ into the above equation, we have

$$\begin{aligned} A^{\mathrm{T}}(t)A(t)\dot{\tilde{X}}(t) + (\dot{A}^{\mathrm{T}}(t)A(t) + A^{\mathrm{T}}(t)\dot{A}(t))\tilde{X}(t) &= A^{\mathrm{T}}(t)A(t)\dot{X}(t) \\ &\quad + (\dot{A}^{\mathrm{T}}(t)A(t) + A^{\mathrm{T}}(t)\dot{A}(t))X(t) - \dot{A}^{\mathrm{T}}(t). \end{aligned}$$

Using ZD model equation (8.4), with $X(t) = \tilde{X}(t) + A^+(t)$, we know that $\tilde{X}(t)$ is the solution to the ensuing dynamics:

$$A^{\mathrm{T}}(t)A(t)\dot{\tilde{X}}(t) + (\dot{A}^{\mathrm{T}}(t)A(t) + A^{\mathrm{T}}(t)\dot{A}(t))\tilde{X}(t) = -\gamma(A^{\mathrm{T}}(t)A(t)\tilde{X}(t)).$$

Because $E(t) = A^{\mathrm{T}}(t)A(t)\tilde{X}(t)$, the above equation is thus rewritten as $\dot{E}(t) = -\gamma(E(t))$, which is a compact matrix form of the following set of $n \times m$ equations:

$$\dot{e}_{ij}(t) = -\gamma e_{ij}(t) \in \mathbb{R},\ \forall i \in \{1, 2, \ldots, n\} \text{ and } j \in \{1, 2, \ldots, m\}. \tag{8.5}$$

Then, we can define a Lyapunov function candidate $v_{ij}(t) = e_{ij}^2(t)/2 \geqslant 0$ for the ijth subsystem (8.5), with its time derivative being

$$\dot{v}_{ij}(t) = \frac{\mathrm{d}v_{ij}(t)}{\mathrm{d}t} = e_{ij}(t)\dot{e}_{ij}(t) = -\gamma e_{ij}^2(t) \leqslant 0.$$

Evidently, $\dot{v}_{ij}(t)$ is negative-definite, i.e., $\dot{v}_{ij}(t) < 0$ for $e_{ij}(t) \neq 0$ and $\dot{v}_{ij}(t) = 0$ for $e_{ij}(t) = 0$. By the Lyapunov stability theory, $e_{ij}(t)$ globally converges to zero for any $i \in \{1, 2, \ldots, n\}$ and $j \in \{1, 2, \ldots, m\}$. In view of $E(t) = A^{\mathrm{T}}(t)A(t)X(t) - A^{\mathrm{T}}(t)$ and the nonsingularity of $A^{\mathrm{T}}(t)A(t)$, we obtain $X(t) \to (A^{\mathrm{T}}(t)A(t))^{-1}A^{\mathrm{T}}(t) \in \mathbb{R}^{n\times m}$ (with $m > n$) as $t \to \infty$. Therefore, based on Theorem 8.1, the state matrix $X(t)$, starting from any initial state $X(0)$, converges to the theoretical time-varying left pseudoinverse $A^+(t)$ of matrix $A(t)$. The proof is thus complete. □

8.2.2 The Second ZF and ZD Model

Inspired by Theorem 8.2, we define the second ZF as

$$E(t) = X(t)(A(t)A^{\mathrm{T}}(t) + \mu I) - A^{\mathrm{T}}(t) \in \mathbb{R}^{n\times m}. \tag{8.6}$$

Then, we obtain the following model by expanding the ZD design formula (7.3):

$$\begin{aligned}\dot{X}(t)(A(t)A^{\mathrm{T}}(t) + \mu I) + X(t)(\dot{A}(t)A^{\mathrm{T}}(t) + A(t)\dot{A}^{\mathrm{T}}(t)) - \dot{A}^{\mathrm{T}}(t)&\\ = -\gamma(X(t)(A(t)A^{\mathrm{T}}(t) + \mu I) - A^{\mathrm{T}}(t)),&\end{aligned}$$

and equivalently

$$\begin{aligned}\dot{X}(t)(A(t)A^{\mathrm{T}}(t) + \mu I) = \dot{A}^{\mathrm{T}}(t) - X(t)(\dot{A}(t)A^{\mathrm{T}}(t) + A(t)\dot{A}^{\mathrm{T}}(t))&\\ - \gamma(X(t)(A(t)A^{\mathrm{T}}(t) + \mu I) - A^{\mathrm{T}}(t)),&\end{aligned} \tag{8.7}$$

Note that the parameter μ should be set appropriately small, in other words, μ should be sufficiently close to 0. Thus, based on ZF (8.6), the ZD model (8.7) is developed for time-varying matrix left pseudoinversion. For such a ZD model, we have the following theorem about its convergence performance.

Theorem 8.4 *Given a smoothly time-varying full-rank matrix $A(t) \in \mathbb{R}^{m\times n}$ with $m > n$, the state matrix $X(t)$ of ZD model (8.7), starting from any initial state $X(0)$, converges to the theoretical time-varying left pseudoinverse $A^{+}(t)$ of matrix $A(t)$.*

Proof Because ZD model (8.7) is derived by using the ZD design method [with the procedure being similar to ZD model (8.4)], we also have

$$\dot{e}_{ij}(t) = -\gamma e_{ij}(t) \in \mathbb{R},\ \forall i \in \{1, 2, \ldots, n\} \text{ and } j \in \{1, 2, \ldots, m\},$$

for ZF (8.6). The proof can thus be generalized from the proof of Theorem 8.3 by considering the initial value of ZF (8.6) [i.e., $X(0)(A(0)A^{\mathrm{T}}(0) + \mu I) - A^{\mathrm{T}}(0)$] and Theorem 8.2. Therefore, such a proof is omitted and is left to interested readers to complete as a topic of exercise. □

8.2.3 The Third ZF and ZD Model

To solve for the time-varying matrix left pseudoinverse, the simplified matrix-valued ZF is defined as

$$E(t) = A(t)X(t) - I \in \mathbb{R}^{m\times m}. \tag{8.8}$$

In view of ZF (8.8), by using the ZD design formula (7.3), we obtain

$$A(t)\dot{X}(t) + \dot{A}(t)X(t) = -\gamma(A(t)X(t) - I),$$

and then

$$A(t)\dot{X}(t) = -\dot{A}(t)X(t) - \gamma(A(t)X(t) - I). \tag{8.9}$$

For the purpose of computation and simulation, left multiplying $A^{\mathrm{T}}(t)$ in both sides of (8.9) yields

$$A^{\mathrm{T}}(t)A(t)\dot{X}(t) = -A^{\mathrm{T}}(t)\dot{A}(t)X(t) - \gamma(A^{\mathrm{T}}(t)A(t)X(t) - A^{\mathrm{T}}(t)). \tag{8.10}$$

Thus, based on ZF (8.8), the resultant ZD model (8.10) is obtained for time-varying matrix left pseudoinversion.

8.2.4 The Fourth ZF and ZD Model

For solving the problem of time-varying matrix left pseudoinverse, another simplified ZF [being different from ZF (8.8)] is defined as

$$E(t) = X(t)A(t) - I \in \mathbb{R}^{n\times n}. \tag{8.11}$$

Similar to the derivation of ZD model (8.10), using formula (7.3), we have

$$\dot{X}(t)A(t) + X(t)\dot{A}(t) = -\gamma(X(t)A(t) - I),$$

and equivalently

$$\dot{X}(t)A(t) = -X(t)\dot{A}(t) - \gamma(X(t)A(t) - I). \tag{8.12}$$

Then, to make Eq. (8.12) more computable, we right multiply $A^{\mathrm{T}}(t)$ in both sides of (8.12), and further obtain

$$\dot{X}(t)A(t)A^{\mathrm{T}}(t) = -X(t)\dot{A}(t)A^{\mathrm{T}}(t) - \gamma(X(t)A(t)A^{\mathrm{T}}(t) - A^{\mathrm{T}}(t)). \tag{8.13}$$

Note that, in (8.13), $A(t)A^{\mathrm{T}}(t)$ is singular (in view of $m > n$). Hence, we adopt the Tikhonov regularization method for (8.13), i.e., add a bias term μI with $\mu \to 0$ to $A(t)A^{\mathrm{T}}(t)$. As a result, (8.13) is rewritten as

$$\dot{X}(t)(A(t)A^{\mathrm{T}}(t) + \mu I) = -X(t)\dot{A}(t)A^{\mathrm{T}}(t) - \gamma(X(t)(A(t)A^{\mathrm{T}}(t) + \mu I) - A^{\mathrm{T}}(t)). \tag{8.14}$$

Thus, based on ZF (8.11), the resultant ZD model (8.14) is obtained to solve for time-varying matrix left pseudoinverse.

8.2.5 The Fifth ZF and ZD Model

By denoting $X^{+}(t) \in \mathbb{R}^{m\times n}$ the time-varying right pseudoinverse of $X(t)$, we can present a new simplified ZF as

$$E(t) = A(t) - X^{+}(t) \in \mathbb{R}^{m\times n}. \tag{8.15}$$

Following the ZD design formula (7.3), we obtain

$$\dot{A}(t) - \dot{X}^{+}(t) = -\gamma(A(t) - X^{+}(t)).$$

By employing Corollary 5.1, the above equation can be further modified as

$$X^{+}(t)\dot{X}(t)X^{+}(t) = -\dot{A}(t) - \gamma(A(t) - X^{+}(t)).$$

Reformulating the above equation, we have

$$\dot{X}(t) = -X(t)\dot{A}(t)X(t) - \gamma(X(t)A(t)X(t) - X(t)). \tag{8.16}$$

Therefore, based on ZF (8.15), we obtain ZD model (8.16), which is exactly the G-M dynamic system for time-varying matrix left pseudoinversion [16–18]. In other words, a direct link between the ZD model and others' model/method (i.e., the G-M dynamic system) for time-varying matrix left pseudoinversion is found. According to Theorem 3.1 of [18], the fundamental G-M dynamic system is locally exponentially convergent rather than globally exponentially convergent. Thus, the initial state of ZD model (8.16) (or say, the G-M dynamic system) should be chosen as $X(0) \approx A^{+}(0)$, i.e., $X(0)$ should be sufficiently close to $A^{+}(0)$.

In summary, defining five different ZFs [i.e., ZFs (8.3), (8.6), (8.8), (8.11), and (8.15)], we have obtained five different ZD models [i.e., (8.4), (8.7), (8.10), (8.14), and (8.16)] to solve for time-varying matrix left pseudoinverse. For readers' convenience and also for comparison purpose, we summarize these ZFs and the corresponding ZD models in Table 8.1. As seen from the table, there exist clear differences among such ZD models. Specifically, the ZFs (termed also the error functions), dynamic equations, and model complexities differ from each other. In practical the practitioner could find and choose the most suitable ZF and the corresponding ZD model in accordance with the specific request.

Table 8.1 Different ZFs resulting in different ZD models for time-varying matrix left pseudoinversion

ZF	ZD model
(8.3)	$A^{\mathrm{T}}(t)A(t)\dot{X}(t)=\dot{A}^{\mathrm{T}}(t)-(\dot{A}^{\mathrm{T}}(t)A(t)+A^{\mathrm{T}}(t)\dot{A}(t))X(t)-\gamma(A^{\mathrm{T}}(t)A(t)X(t)-A^{\mathrm{T}}(t))$
(8.6)	$\dot{X}(t)(A(t)A^{\mathrm{T}}(t)+\mu I)=$ $\dot{A}^{\mathrm{T}}(t)-X(t)(\dot{A}(t)A^{\mathrm{T}}(t)+A(t)\dot{A}^{\mathrm{T}}(t))-\gamma(X(t)(A(t)A^{\mathrm{T}}(t)+\mu I)-A^{\mathrm{T}}(t))$
(8.8)	$A^{\mathrm{T}}(t)A(t)\dot{X}(t)=-A^{\mathrm{T}}(t)\dot{A}(t)X(t)-\gamma(A^{\mathrm{T}}(t)A(t)X(t)-A^{\mathrm{T}}(t))$
(8.11)	$\dot{X}(t)(A(t)A^{\mathrm{T}}(t)+\mu I)=-X(t)\dot{A}(t)A^{\mathrm{T}}(t)-\gamma(X(t)(A(t)A^{\mathrm{T}}(t)+\mu I)-A^{\mathrm{T}}(t))$
(8.15)	$\dot{X}(t)=-X(t)\dot{A}(t)X(t)-\gamma(X(t)A(t)X(t)-X(t))$

8.3 Illustrative Examples

In the previous sections, five different ZD models based on different ZFs have been proposed for time-varying matrix left pseudoinversion. In this section, three illustrative examples are provided for substantiating the efficacy of such five ZD models on solving for time-varying matrix left pseudoinverse.

Example 8.1 Let us consider the following time-varying full-rank matrix $A(t)$:

$$A(t)=\begin{bmatrix}\sin(2t) & \cos(2t)\\ -\cos(2t) & \sin(2t)\\ \sin(2t) & \cos(2t)\end{bmatrix}\in\mathbb{R}^{3\times 2}. \tag{8.17}$$

Then, according to Eq. (8.1), the theoretical time-varying left pseudoinverse of matrix A in (8.17) is obtained as

$$A^{+}(t)=\begin{bmatrix}0.5\sin(2t) & -\cos(2t) & 0.5\sin(2t)\\ 0.5\cos(2t) & \sin(2t) & 0.5\cos(2t)\end{bmatrix}\in\mathbb{R}^{2\times 3}.$$

Because we have got theoretical time-varying pseudoinverse $A^{+}(t)$, we can use it as a criterion to check the efficacy of the proposed ZD models (8.4) and (8.7). The corresponding simulation results are shown in Figs. 8.1, 8.2, 8.3, and 8.4.

First, we investigate the convergence performance of ZD model (8.4) using $\gamma=1$, with the related simulations results shown in Figs. 8.1 and 8.2. As illustrated in Fig. 8.1, starting from a randomly-generated initial state $X(0)\in\mathbb{R}^{2\times 3}$, state matrix $X(t)\in\mathbb{R}^{2\times 3}$ of ZD model (8.4) converges to the theoretical time-varying pseudoinverse $A^{+}(t)$ of matrix $A(t)$ in (8.17) accurately and rapidly. In addition, we show the residual error $\|E(t)\|_{\mathrm{F}}=\|A^{\mathrm{T}}(t)A(t)X(t)-A^{\mathrm{T}}(t)\|_{\mathrm{F}}$ synthesized by ZD model (8.4) in Fig. 8.2. It follows from Fig. 8.2 that residual error via (8.4) diminishes exponentially to zero within around 6 s. Thus, based on these simulation results, the efficacy of ZD model (8.4) for time-varying matrix left pseudoinversion is substantiated.

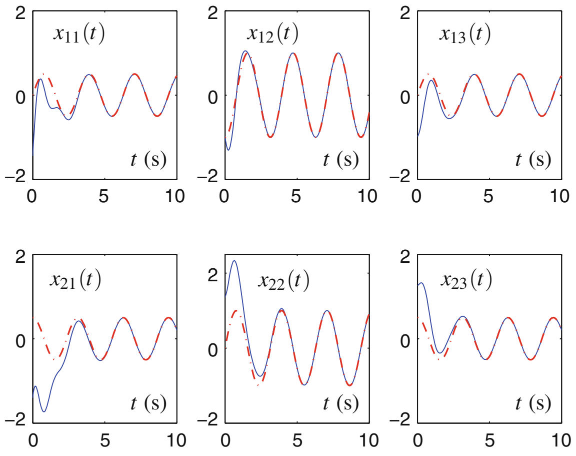


Fig. 8.1 State trajectories of ZD model (8.4), where *dash-dotted curves* denote theoretical pseudoinverse $A^+(t)$ of matrix $A(t)$ in (8.17) and *solid curves* denote the neural-state solution

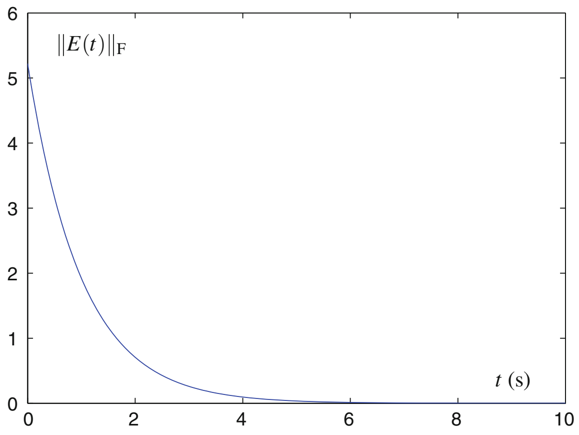


Fig. 8.2 Residual error $\|E(t)\|_F = \|A^T(t)A(t)X(t) - A^T(t)\|_F$ synthesized by ZD model (8.4) for the pseudoinverse of matrix $A(t)$ in (8.17)

Second, we investigate the convergence performance of ZD model (8.7) using $\gamma = 1$ and $\mu = 10^{-3}$. For substantiating the effectiveness of ZD model (8.7), the neural-state matrix $X(t) \in \mathbb{R}^{2\times 3}$ is shown in Fig. 8.3. As seen from such a figure, $X(t)$ converges to the theoretical time-varying pseudoinverse $A^+(t)$ of matrix $A(t)$ in (8.17) rapidly. In addition, Fig. 8.4 further shows the corresponding transient behavior of $\|E(t)\|_F$ synthesized by ZD model (8.7). Evidently, from Fig. 8.4, we know that the residual error via (8.7) converges exponentially to zero within around 6 s. These simulation results substantiate the efficacy of ZD model (8.7) for time-varying matrix left pseudoinverse computation.

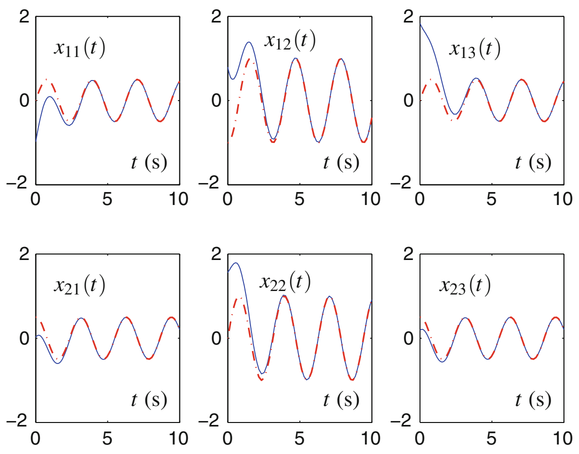


Fig. 8.3 State trajectories of ZD model (8.7), where *dash-dotted curves* denote theoretical pseudoinverse $A^+(t)$ of matrix $A(t)$ in (8.17) and *solid curves* denote the neural-state solution

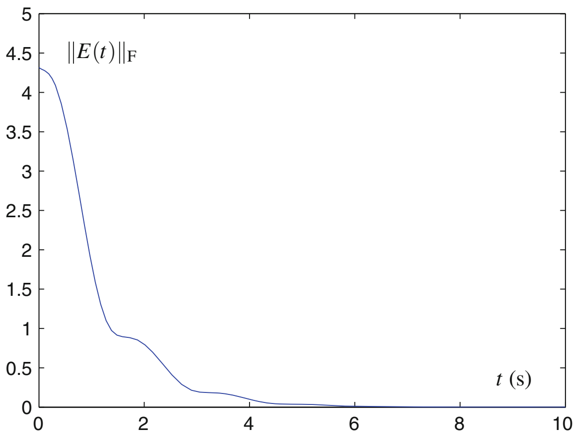


Fig. 8.4 Residual error $\|E(t)\|_\text{F} = \|A^\text{T}(t)A(t)X(t) - A^\text{T}(t)\|_\text{F}$ synthesized by ZD model (8.7) for the pseudoinverse of matrix $A(t)$ in (8.17)

Example 8.2 For further substantiating the efficacy of ZD models (8.10), (8.14), and (8.16), let us consider the following time-varying full-rank matrix $A(t)$:

$$A(t) = \begin{bmatrix} \sin(3t) & \cos(3t) \\ -\cos(3t) & \sin(3t) \\ \sin(3t) & \cos(3t) \end{bmatrix} \in \mathbb{R}^{3\times 2}. \tag{8.18}$$

It follows (8.1) that the theoretical time-varying left pseudoinverse of matrix $A(t)$ in (8.18) is

$$A^+(t) = \begin{bmatrix} 0.5\sin(3t) & -\cos(3t) & 0.5\sin(3t) \\ 0.5\cos(3t) & \sin(3t) & 0.5\cos(3t) \end{bmatrix} \in \mathbb{R}^{2\times 3}. \tag{8.19}$$

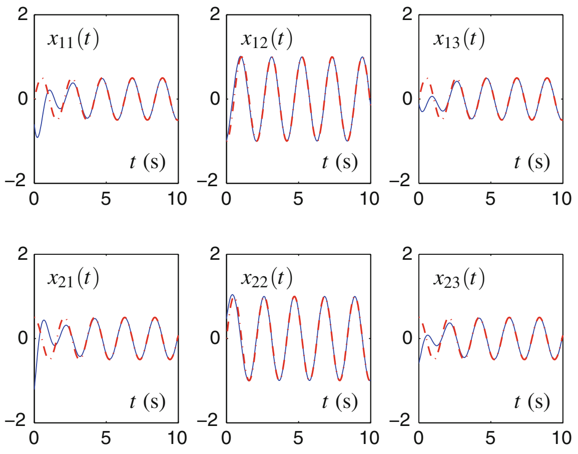


Fig. 8.5 State trajectories of ZD model (8.10), where *dash-dotted curves* denote theoretical pseudoinverse $A^+(t)$ of matrix $A(t)$ in (8.18) and *solid curves* denote the neural-state solution

Note that the variation frequency of $A(t)$ in (8.18) is one and a half times that of Example 8.3. The corresponding simulation results are shown in Figs. 8.5 and 8.6.

Figure 8.5 illustrates the state matrix $X(t)$ of ZD model (8.10) with $\gamma = 1$. Note that the state matrices $X(t)$ synthesized by ZD models (8.14) and (8.16) are not presented again because they are similar with that shown in Fig. 8.5. As seen from Fig. 8.5, starting from a randomly-generated initial state $X(0) \in \mathbb{R}^{2\times 3}$, state matrix $X(t) \in \mathbb{R}^{2\times 3}$ of ZD model (8.10) converges to the theoretical time-varying pseudoinverse $A^+(t)$ of matrix $A(t)$ in (8.18) in a short time. Besides, Fig. 8.6 shows the transient behavior of $\|E(t)\|_\text{F}$ synthesized by ZD models (8.10), (8.14), and (8.16). From Fig. 8.6, we can see that residual errors $\|E(t)\|_\text{F}$ of such three ZD models (8.10), (8.14), and (8.16) all converge to zero. Therefore, the efficacy of ZD models (8.10), (8.14), and (8.16) for the time-varying pseudoinverse is also substantiated.

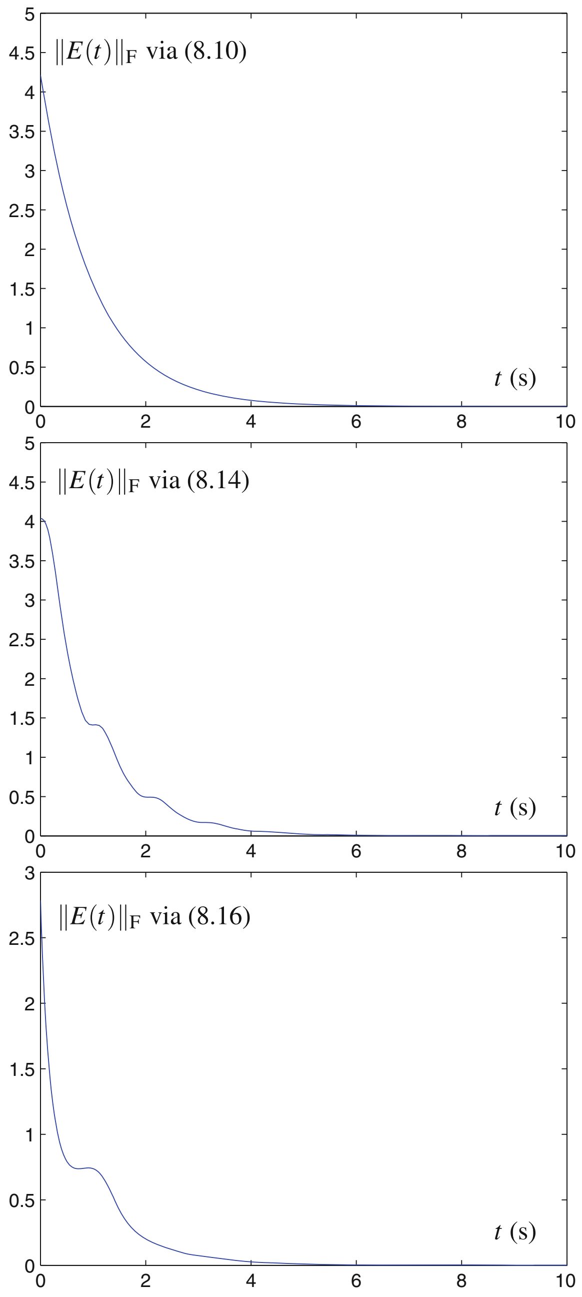
Example 8.3 In this example, we consider a more complicated time-varying full-rank matrix $A(t)$ as follows:

$$A(t) = \begin{bmatrix} a_{11}(t) & a_{12}(t) & a_{13}(t) & \cdots & a_{16}(t) \\ a_{21}(t) & a_{22}(t) & a_{23}(t) & \cdots & a_{26}(t) \\ a_{31}(t) & a_{32}(t) & a_{33}(t) & \cdots & a_{36}(t) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{71}(t) & a_{72}(t) & a_{73}(t) & \cdots & a_{76}(t) \end{bmatrix} \in \mathbb{R}^{7\times 6}, \tag{8.20}$$

where the element $a_{ij}(t)$ is set as

$$a_{ij}(t) = \begin{cases} \sin(t), & \text{if } i = j, \\ i + \cos(t), & \text{if } i > j, \\ j + \cos(t), & \text{if } i < j. \end{cases}$$

Fig. 8.6 Residual errors $\|E(t)\|_\mathrm{F} = \|A^\mathrm{T}(t)A(t)X(t) - A^\mathrm{T}(t)\|_\mathrm{F}$ synthesized by ZD models (8.10), (8.14), and (8.16) for the pseudoinverse of matrix $A(t)$ in (8.18)



Note that, because of the complexity of matrix $A(t)$ in (8.20), the analytical theoretical pseudoinverse solution is difficult to be obtained. In addition, due to results similarity, we only present the convergence performance of the residual errors $\|E(t)\|_{\mathrm{F}}$ synthesized by ZD models (8.4) and (8.10) with different γ values in this example. The corresponding simulation results are shown in Figs. 8.7 and 8.8. As seen from Figs. 8.7 and 8.8, starting from a randomly-generated initial state, the residual error $\|E(t)\|_{\mathrm{F}}$ synthesized by (8.4) or (8.10) diminishes to zero, which means that the corresponding solutions $X(t)$ converge to the theoretical time-varying pseudoinverse of matrix $A(t)$ in (8.20) rapidly and accurately. Thus, the efficacy of the proposed ZD models (8.4) and (8.10) on solving for the more complicated time-varying pseudoinverse is also substantiated evidently. Moreover, comparing Fig. 8.7 with Fig. 8.8, we observe that the convergence time of residual errors $\|E(t)\|_{\mathrm{F}}$ becomes shorter as the value of γ increases (showing the important role of γ). Specifically, the convergence time corresponding to (8.4) or (8.10) can be expedited from around 0.7–0.07 s, when the γ value is increased from 10 to 100. This observation tells that ZD models (8.4) and (8.10) both have the exponential convergence property, which can be

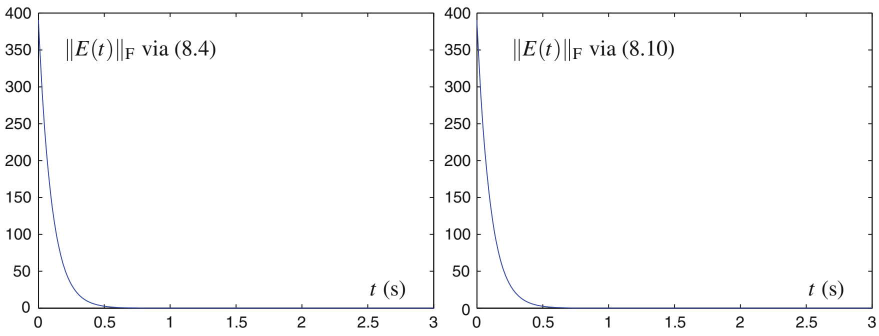


Fig. 8.7 Residual errors $\|E(t)\|_{\mathrm{F}} = \|A^{\mathrm{T}}(t)A(t)X(t) - A^{\mathrm{T}}(t)\|_{\mathrm{F}}$ synthesized by ZD models (8.4) and (8.10) with $\gamma = 10$ for the pseudoinverse of matrix $A(t)$ in (8.20)

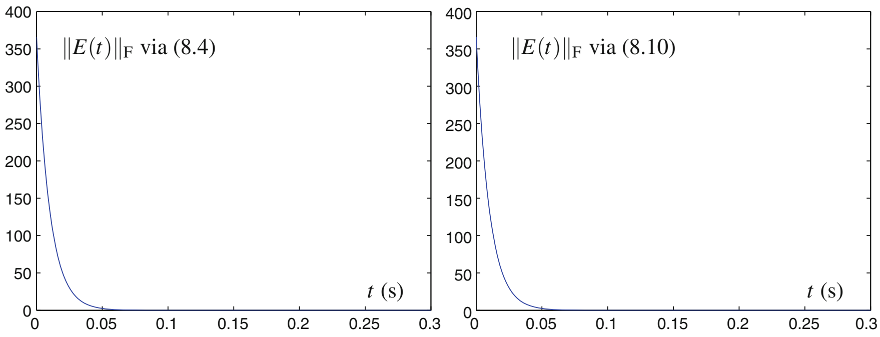


Fig. 8.8 Residual errors $\|E(t)\|_{\mathrm{F}} = \|A^{\mathrm{T}}(t)A(t)X(t) - A^{\mathrm{T}}(t)\|_{\mathrm{F}}$ synthesized by ZD models (8.4) and (8.10) with $\gamma = 100$ for the pseudoinverse of matrix $A(t)$ in (8.20)

expedited effectively by increasing the value of γ. Note that, for other ZD models [i.e., (8.7), (8.14), and (8.16)], we have the same or similar observations which are omitted here due to results similarity. In addition, based on the above three illustrative examples, we can conclude that the convergence time of the proposed ZD models does not increase as the matrix dimension increases. Besides, being a topic of exercise, the corresponding simulative verifications of ZD models (8.7), (8.14), and (8.16) are left for interested readers.

In summary, from the above three computer-simulation examples, we have substantiated the efficacy of the proposed ZD models (8.4), (8.7), (8.10), (8.14), and (8.16) for time-varying matrix left pseudoinversion.

8.4 Summary

In this chapter, by defining five different ZFs (8.3), (8.6), (8.8), (8.11), and (8.15), five different ZD models (8.4), (8.7), (8.10), (8.14), and (8.16) have been proposed, generalized, developed, and investigated to solve for time-varying matrix left pseudoinverse. In addition, the relationship between ZD model (8.16) and G-M dynamic system for time-varying matrix left pseudoinversion has been discovered. Computer-simulation results with three illustrative examples have further substantiated the efficacy of the proposed ZD models for time-varying matrix left pseudoinverse computation.

References

1. Juang LH, Wu MN (2010) Image noise reduction using Wiener filtering with pseudo-inverse. Measurement 43(10):1649–1655
2. Van der Veen AJ, Talwar S, Paulraj A (1997) A subspace approach to blind space-time signal processing for wireless communication systems. IEEE Trans Signal Process 45(1):173–190
3. Lin J, Lin CC, Lo HS (2009) Pseudo-inverse Jacobian control with grey relational analysis for robot manipulators mounted on oscillatory bases. J Sound Vib 326(3–5):421–437
4. Skurichina M, Duin RP (2002) Bagging, boosting and the random subspace method for linear classifiers. Pattern Anal Appl 5(2):121–135
5. Zhang BL, Zhang H, Ge SS (2004) Face recognition by applying wavelet subband representation and kernel associative memory. IEEE Trans Neural Netw 15(1):166–177
6. Perković MD, Stanimirović PS (2011) Iterative method for computing the Moore-Penrose inverse besed on Penrose equations. J Comput Appl Math 235(6):1604–1613
7. Courrieu P (2005) Fast computation of Moore-Penrose inverse matrices. Neural Inf Process Lett Rev 8(2):25–29
8. Guo W, Huang T (2010) Method of elementary transformation to compute Moore-Penrose inverse. Appl Math Comput 216(5):1614–1617
9. Tasić MB, Stanimirović PS, Petković MD (2007) Symbolic computation of weighted Moore-Penrose inverse using partitioning method. Appl Math Comput 189(1):615–640
10. Zhang Y, Yi C (2011) Zhang neural networks and neural-dynamic method. Nova Science Publishers, New York

11. Zhang Y, Yang Y, Tan N, Cai B (2011) Zhang neural network solving for time-varying full-rank matrix Moore-Penrose inverse. Computing 92(2):97–121
12. Guo D, Zhang Y (2014) Li-function activated ZNN with finite-time convergence applied to redundant-manipulator kinematic control via time-varying Jacobian matrix pseudoinversion. Appl Soft Comput 24:158–168
13. Liao B, Zhang Y (2014) From different ZFs to different ZNN models accelerated via Li activation functions to finite-time convergence for time-varying matrix pseudoinversion. Neurocomputing 133:512–522
14. Ben-Israel A, Greville TNE (2003) Generalized inverses: theory and applications, 2nd edn. Springer, New York
15. Wang J (1997) Recurrent neural networks for computing pseudoinverses of rank-deficient matrices. SIAM J Sci Comput 19(5):1479–1493
16. Getz NH, Marsden JE (1995) A dynamic inverse for nonlinear maps. In: Proceedings of 34th IEEE conference on decision and control, pp 4218–4223
17. Getz NH, Marsden JE (1995) Joint-space tracking of workspace trajectories in continuous time. In: Proceedings of 34th IEEE conference on decision and control, pp 1001–1006
18. Getz NH, Marsden JE (1997) Dynamical methods for polar decomposition and inversion of matrices. Linear Algebra Appl 258:311–343

Chapter 9
Time-Varying Matrix Right Pseudoinverse

Abstract In Chap. 8, different ZD models based on ZFs have been presented for time-varying matrix left pseudoinversion. Being another case study of pseudoinverse for a time-varying rectangular matrix, in this chapter, by introducing four different ZFs, four different ZD models are proposed, generalized, developed, and investigated for time-varying right pseudoinversion. In addition, the link between the ZD models and the Getz-Marsden (G-M) dynamic system is discovered and presented to solve for time-varying matrix right pseudoinverse. Theoretical results and computer simulations with three illustrative examples are provided to further substantiate the excellent convergence performance of the proposed ZD models for time-varying matrix right pseudoinversion.

9.1 Introduction

As presented in Chap. 8, there are two cases of time-varying pseudoinverse $A^+(t) \in \mathbb{R}^{n\times m}$ for matrix $A(t) \in \mathbb{R}^{m\times n}$ with $m \neq n$; i.e., the left pseudoinverse (corresponding to $m > n$) and the right pseudoinverse (corresponding to $m < n$). In addition, different ZD models based on ZFs have been presented for time-varying matrix left pseudoinversion in Chap. 8. Thus, in this chapter, we focus on solving for time-varying right pseudoinverse of matrix $A(t) \in \mathbb{R}^{m\times n}$ with $m < n$.

Note that the problem of solving for (time-varying) matrix right pseudoinverse is one of the basic issues encountered in a large number of science and engineering fields [1–5]. For example, the pseudoinverse-type solutions have been applied to motion planning of redundant robot manipulators [6–8]. Due to the important role of the right pseudoinverse, much effort has been contributed to fast solution of right pseudoinverse, and subsequently a number of algorithms/methods (including those ZD models presented in Chap. 8) [6, 9–16] have been developed and studied for static and/or time-varying matrix right pseudoinversion.

In this chapter, focusing on time-varying matrix right pseudoinversion, we propose, generalize, develop, and investigate four different ZD models by introducing four different ZFs. In addition, the link between the ZD models and the Getz-Marsden (G-M) dynamic system [11–13] is presented to solve for time-varying matrix right pseudoinverse. Theoretical results are also provided, which show that the

© Springer-Verlag Berlin Heidelberg 2015
Y. Zhang and D. Guo, *Zhang Functions and Various Models*,
DOI 10.1007/978-3-662-47334-4_9

proposed ZD models have the global/exponential convergence performance. Through computer simulations with three illustrative examples, we further substantiate the efficacy of the proposed ZD models for time-varying matrix right pseudoinversion.

9.2 ZFs and ZD Models

In this section, we introduce four different ZFs and propose the resultant ZD models for time-varying matrix right pseudoinversion (together with the link to the G-M dynamic system).

For a given time-varying full-rank matrix $A(t) \in \mathbb{R}^{m\times n}$ with $m < n$, based on Theorem 8.1, the unique time-varying right pseudoinverse $A^+(t)$ of $A(t)$ is given as

$$A^+(t) = A^{\mathrm{T}}(t)(A(t)A^{\mathrm{T}}(t))^{-1} \in \mathbb{R}^{n\times m}. \tag{9.1}$$

Then, based on (9.1) and Definition 8.1, we know that matrix $A^+(t)$ satisfies the following equations (being a basis for defining different ZFs):

$$A^+(t)A(t)A^{\mathrm{T}}(t) = A^{\mathrm{T}}(t) \in \mathbb{R}^{n\times m} \text{ and } A(t)A^+(t) = I \in \mathbb{R}^{m\times m}.$$

By denoting $X(t) \in \mathbb{R}^{n\times m}$ as the unknown matrix to be obtained, this chapter aims at designing different ZFs to construct various ZD models to solve for the time-varying matrix right pseudoinverse. That is to say, the unknown $X(t)$ can be obtained by using ZD models in real-time t such that it converges to the exact theoretical time-varying right pseudoinverse $A^+(t)$.

Specifically, to solve for time-varying matrix right pseudoinverse, in this chapter, we define four different ZFs as follows:

$$E(t) = A(t)X(t) - I, \tag{9.2}$$
$$E(t) = A(t) - X^+(t), \tag{9.3}$$
$$E(t) = X(t)A(t)A^{\mathrm{T}}(t) - A^{\mathrm{T}}(t), \tag{9.4}$$
$$E(t) = X(t)A(t) - I, \tag{9.5}$$

According to the ZD model design formula (7.3), based on four different ZFs [i.e., (9.2)–(9.5)], four different ZD models for time-varying matrix right pseudoinversion are thus derived and presented as follows.

- Combining the ZD design formula (7.3) and ZF (9.2), we have

$$\dot{A}(t)X(t) + A(t)\dot{X}(t) = -\gamma(A(t)X(t) - I),$$

which is rewritten as

$$A(t)\dot{X}(t) = -\dot{A}(t)X(t) - \gamma(A(t)X(t) - I), \tag{9.6}$$

Then, to make (9.6) more computable, we left multiply $A^{\mathrm{T}}(t)$ on both sides of (9.6), and further obtain

$$A^{\mathrm{T}}(t)A(t)\dot{X}(t) = -A^{\mathrm{T}}(t)\dot{A}(t)X(t) - \gamma\left(A^{\mathrm{T}}(t)A(t)X(t) - A^{\mathrm{T}}(t)\right). \tag{9.7}$$

It is worth noting that $A^{\mathrm{T}}(t)A(t)$ is singular (in view of $m < n$). Hence, ZD model (9.7) cannot be used directly to solve for time-varying matrix right pseudoinverse $A^{+}(t)$. In order to make ZD model (9.7) computable and make $X(t)$ converge to the unique solution, we can adopt the Tikhonov regularization method for (9.7), i.e., add a bias term μI with $\mu \to 0$ to $A^{\mathrm{T}}(t)A(t)$. Therefore, based on the above analysis, ZD model (9.7) is modified as

$$(A^{\mathrm{T}}(t)A(t)+\mu I)\dot{X}(t) = -A^{\mathrm{T}}(t)\dot{A}(t)X(t)-\gamma\left((A^{\mathrm{T}}(t)A(t)+\mu I)X(t) - A^{\mathrm{T}}(t)\right). \tag{9.8}$$

Thus, we obtain the final ZD model (9.8) based on ZF (9.2) for time-varying matrix right pseudoinversion. Additionally, design parameter μ should be set appropriately small for the convergence of ZD model (9.8) to the solution.

- By considering ZF (9.3) and following the ZD design formula (7.3), the time derivative of $E(t)$ is obtained with a minimum-norm derivation [see also Corollary 5.1 for the derivation of $\dot{X}^{+}(t)$]:

$$\dot{E}(t) = \dot{A}(t) - \dot{X}^{+}(t) = \dot{A}(t) + X^{+}(t)\dot{X}(t)X^{+}(t) = -\gamma E(t) = -\gamma\left(A(t) - X^{+}(t)\right).$$

Reformulating the above equation (with $X(t)X^{+}(t)$ replaced by I), we have the following new ZD model aiming at solving for the time-varying matrix right pseudoinverse:

$$\dot{X}(t) = -X(t)\dot{A}(t)X(t) - \gamma(X(t)A(t)X(t) - X(t)). \tag{9.9}$$

Thus, ZD model (9.9) based on ZF (9.3) is obtained. Note that such a ZD model (9.9) is also the G-M dynamic system [11–13] for time-varying matrix right pseudoinversion. In other words, the G-M dynamic system is found to be a special case of the ZD models. In addition, such a G-M dynamic system requires the initial state $X(0)$ to be close enough to the theoretical initial right pseudoinverse $A^{+}(0)$.

- Combining the ZD design formula (7.3) and ZF (9.4), we have

$$\begin{aligned}\dot{X}(t)A(t)A^{\mathrm{T}}(t) = &-X(t)\left(\dot{A}(t)A^{\mathrm{T}}(t) + A(t)\dot{A}^{\mathrm{T}}(t)\right)\\ &+ \dot{A}^{\mathrm{T}}(t) - \gamma\left(X(t)A(t)A^{\mathrm{T}}(t) - A^{\mathrm{T}}(t)\right).\end{aligned} \tag{9.10}$$

Therefore, based on ZF (9.4), we have ZD model (9.10) for time-varying matrix right pseudoinversion.

- With the ZD design formula (7.3) and ZF (9.5) combined, we have

$$\dot{X}(t)A(t) = -X(t)\dot{A}(t) - \gamma(X(t)A(t) - I).$$

Table 9.1 Different ZFs resulting in different ZD models for time-varying matrix right pseudoinversion

ZF	ZD model
(9.2)	$(A^{\mathrm{T}}(t)A(t)+\mu I)\dot{X}(t)=-\gamma\left((A^{\mathrm{T}}(t)A(t)+\mu I)X(t)-A^{\mathrm{T}}(t)\right)-A^{\mathrm{T}}(t)\dot{A}(t)X(t)$
(9.3)	$\dot{X}(t)=-\gamma\,(X(t)A(t)X(t)-X(t))-X(t)\dot{A}(t)X(t)$
(9.4)	$\dot{X}(t)A(t)A^{\mathrm{T}}(t)=$ $-\gamma\left(X(t)A(t)A^{\mathrm{T}}(t)-A^{\mathrm{T}}(t)\right)-X(t)\left(\dot{A}(t)A^{\mathrm{T}}(t)+A(t)\dot{A}^{\mathrm{T}}(t)\right)+\dot{A}^{\mathrm{T}}(t)$
(9.5)	$\dot{X}(t)A(t)A^{\mathrm{T}}(t)=-\gamma\left(X(t)A(t)A^{\mathrm{T}}(t)-A^{\mathrm{T}}(t)\right)-X(t)\dot{A}(t)A^{\mathrm{T}}(t)$

To make the above equation computable, we right multiply $A^{\mathrm{T}}(t)$ in both sides, and further obtain

$$\dot{X}(t)A(t)A^{\mathrm{T}}(t)=-X(t)\dot{A}(t)A^{\mathrm{T}}(t)-\gamma\left(X(t)A(t)A^{\mathrm{T}}(t)-A^{\mathrm{T}}(t)\right). \quad (9.11)$$

Therefore, we have ZD model (9.11) based on ZF (9.5) for time-varying matrix right pseudoinversion.

As a result, we have obtained four different ZD models, i.e., ZD models (9.8)–(9.11), corresponding to four different ZFs, i.e., ZFs (9.2)–(9.5). For readers' convenience, we summarize these ZFs and ZD models in Table 9.1.

9.3 Theoretical Results

In this section, we present and compare the following propositions on the convergence characteristics of the proposed ZD models for time-varying matrix right pseudoinversion. Note that the proofs of such propositions can be generalized from that of Theorem 8.3 presented in Chap. 8 (and the related proofs are also left to interested readers to complete as a topic of exercise).

Proposition 9.1 *Consider a smoothly time-varying full-rank matrix $A(t)\in\mathbb{R}^{m\times n}$ with $m<n$. The state matrix $X(t)$ of the theoretical ZD model (9.6) based on ZF (9.2), starting from any initial state $X(0)\in\mathbb{R}^{n\times m}$, converges to the theoretical time-varying right pseudoinverse $A^{+}(t)$ of matrix $A(t)$.*

Proposition 9.2 *Consider a smoothly time-varying full-rank matrix $A(t)\in\mathbb{R}^{m\times n}$ with $m<n$. The state matrix $X(t)$ of the practical ZD model (9.8) based on ZF (9.2), starting from a randomly-generated initial state $X(0)\in\mathbb{R}^{n\times m}$, converges to the theoretical time-varying right pseudoinverse $A^{+}(t)$ of matrix $A(t)$.*

Proposition 9.3 *Consider a smoothly time-varying full-rank matrix $A(t)\in\mathbb{R}^{m\times n}$ with $m<n$. Starting from an initial state $X(0)\in\mathbb{R}^{n\times m}$ which is close enough to*

$A^+(0)$, *the state matrix* $X(t)$ *of ZD model (9.9) based on ZF (9.3) converges to the theoretical time-varying right pseudoinverse* $A^+(t)$ *of matrix* $A(t)$.

Besides, the propositions on the convergence characteristics of ZD models (9.10) and (9.11) have been given and investigated in the other book [15]. Therefore, we do not detail such two ZD models (9.10) and (9.11) in this chapter/book.

9.4 Illustrative Examples

In this section, we have three illustrative examples to substantiate the efficacy of ZD models (9.8) and (9.9) and discuss how parameters γ and μ effect the convergence performance of such two ZD models.

Example 9.1 Let us consider the following time-varying full-rank matrix:

$$A(t) = \begin{bmatrix} \sin(1.5t) & \cos(1.5t) & -\sin(1.5t) \\ -\cos(1.5t) & \sin(1.5t) & \cos(1.5t) \end{bmatrix} \in \mathbb{R}^{2\times 3}, \tag{9.12}$$

of which the theoretical time-varying right pseudoinverse is given as below for comparative purposes (i.e., to check the correctness of ZD solutions):

$$A^+(t) = \begin{bmatrix} 0.5\sin(1.5t) & -0.5\cos(1.5t) \\ \cos(1.5t) & \sin(1.5t) \\ -0.5\sin(1.5t) & 0.5\cos(1.5t) \end{bmatrix} \in \mathbb{R}^{3\times 2}.$$

We exploit ZD models (9.8) and (9.9) to solve for the time-varying right pseudoinverse of matrix $A(t)$ in (9.12) with $\gamma = 10$ and $\mu = 10^{-8}$. In Fig. 9.1, the neural

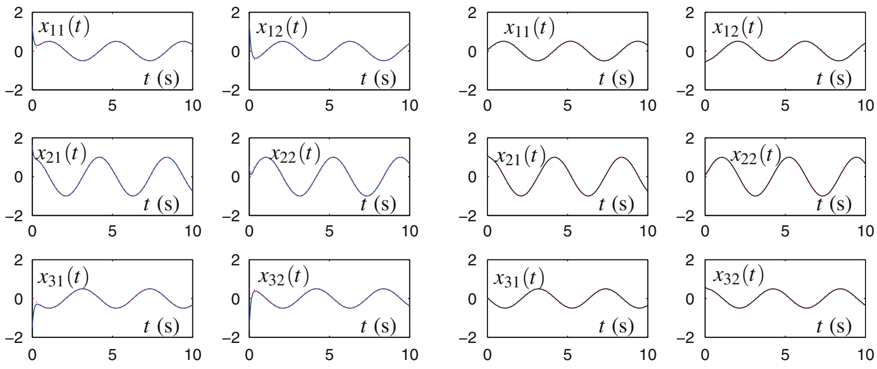


Fig. 9.1 State trajectories of ZD models (9.8) and (9.9) with $\gamma = 10$ and $\mu = 10^{-8}$ solving for the time-varying right pseudoinverse of matrix $A(t)$ in (9.12)

states $X(t)$ of ZD models (9.8) and (9.9) are respectively shown, with the theoretical time-varying right pseudoinverse $A^+(t)$ denoted by the dash-dotted curves. From Fig. 9.1, we see that the state trajectories of ZD model (9.8) always converge to the $A^+(t)$ trajectories. In contrast, there exists appreciable difference between the $A^+(t)$ trajectories and the state trajectories of ZD model (9.9) (which is also the G-M dynamic system). Thus, the difference and efficacy of the proposed ZD models (resulting from different ZFs) are substantiated.

Example 9.2 In this example, we consider a time-varying full-rank matrix as follows:

$$A(t) = \begin{bmatrix} 0.5\sin(t) & -\cos(t) & 0.5\sin(t) \\ 0.5\cos(t) & \sin(t) & 0.5\cos(t) \end{bmatrix} \in \mathbb{R}^{2\times 3}, \tag{9.13}$$

whose theoretical time-varying right pseudoinverse is given as below for comparison:

$$A^+(t) = \begin{bmatrix} \sin(t) & \cos(t) \\ -\cos(t) & \sin(t) \\ \sin(t) & \cos(t) \end{bmatrix} \in \mathbb{R}^{3\times 2}.$$

About ZD model (9.8) solving for the time-varying right pseudoinverse of matrix $A(t)$ in (9.13) with $\mu = 10^{-8}$ and different values of γ, the solution errors $\|X(t) - A^+(t)\|_F$ are displayed in Fig. 9.2. When $\gamma = 10$, the solution error $\|X(t) - A^+(t)\|_F$ converges to zero in about 0.7 s; and when $\gamma = 20$ and $\gamma = 40$, the convergence time of the solution error toward zero is shortened to about 0.35 and 0.15 s, respectively. Therefore, we can set γ appropriately large to expedite the ZD-solution process.

Example 9.3 For comparative purposes, we set the value of μ to be 10^{-4}, 10^{-8} and 10^{-12} in this example to solve for the time-varying right pseudoinverse of matrix $A(t)$ in (9.12) via ZD model (9.8). As shown in Fig. 9.3, the solution errors of ZD model (9.8) (with $\gamma = 10$) all converge to zero rapidly. Though different values of μ

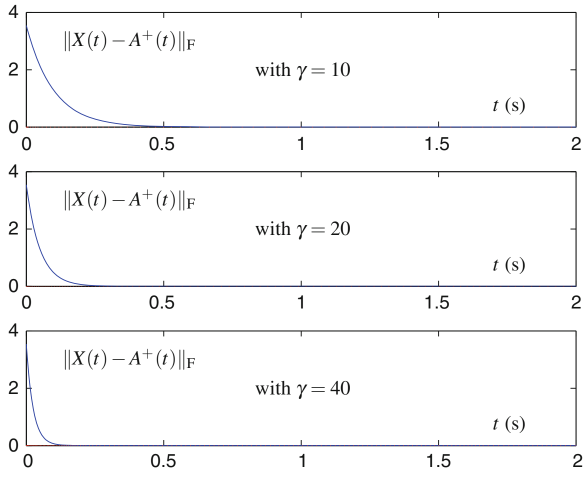


Fig. 9.2 Solution errors $\|X(t) - A^+(t)\|_F$ of ZD model (9.8) solving for the time-varying right pseudoinverse of matrix $A(t)$ in (9.13) with $\mu = 10^{-8}$ fixed and with different values of γ tested (i.e., with $\gamma = 10$, 20 and 40)

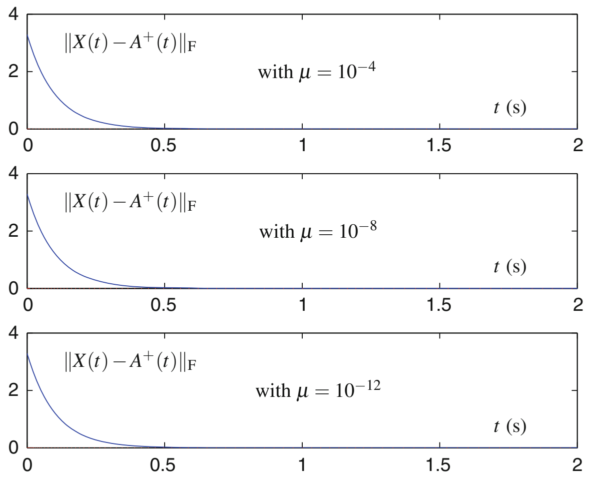


Fig. 9.3 Solution errors $\|X(t)-A^+(t)\|_\mathrm{F}$ of ZD model (9.8) solving for time-varying right pseudoinverse of matrix $A(t)$ in (9.12) with $\gamma=10$ fixed and with different values of μ tested

are used, the convergence time is almost the same. Thus, we come to the conclusion that, when the μ value is set appropriately small, ZD model (9.8) can achieve the same excellent convergence performance, i.e., solving for the time-varying right pseudoinverse accurately.

9.5 Summary

In this chapter, by defining four different ZFs (9.2) through (9.5), four different ZD models (9.8) through (9.11) have been proposed, generalized, developed, and investigated to solve for time-varying matrix right pseudoinverse. In addition, the relationship between ZD model (9.8) and the G-M dynamic system for time-varying matrix right pseudoinversion has been discovered and presented. Theoretical results and computer simulation results with three illustrative examples have been provided to substantiate the efficacy of the proposed ZD models for time-varying matrix right pseudoinverse computation.

References

1. Fieguth PW, Menemenlis D, Fukumori I (2003) Mapping and pseudoinverse algorithms for ocean data assimilation. IEEE Trans Geosci Remote Sens 41(1):43–51
2. Park J, Choi Y, Chung WK, Youm Y (2001) Multiple tasks kinematics using weighted pseudoinverse for kinematically redundant manipulators. In: Proceedings of the IEEE conference on robotics and automation, pp 4041–4047
3. Hu J, Qian S, Ding Y (2010) Improved pseudoinverse algorithm and its application in controlling acoustic field generated by phased array. J Syst Simul 22(5):1111–1116

4. Dean P, Porrill J (1998) Pseudo-inverse control in biological systems: a learning mechanism for fixation stability. Neural Netw 11(7–8):1205–1218
5. Guo P, Lyu MR (2004) A pseudoinverse learning algorithm for feedforward neural networks with stacked generalization applications to software reliability growth data. Neurocomputing 56:101–121
6. Guo D, Zhang Y (2014) Li-function activated ZNN with finite-time convergence applied to redundant-manipulator kinematic control via time-varying Jacobian matrix pseudoinversion. Appl Soft Comput 24:158–168
7. Klein CA, Kee KB (1989) The nature of drift in pseudoinverse control of kinematically redundant manipulators. IEEE Trans Robot Autom 5(2):231–234
8. Zhang Y, Guo D, Ma S (2013) Different-level simultaneous minimization of joint-velocity and joint-torque for redundant robot manipulators. J Intell Robot Syst 72(3–4):301–323
9. Wei Y, Cai J, Ng MK (2004) Computing Moore-Penrose inverses of Toeplitz matrices by Newton's iteration. Math Comput Model 40(1–2):181–191
10. Wang J (1997) Recurrent neural networks for computing pseudoinverses of rank-deficient matrices. SIAM J Sci Comput 19(5):1479–1493
11. Getz NH, Marsden JE (1995) A dynamic inverse for nonlinear maps. In: Proceedings of 34th IEEE conference on decision and control, pp 4218–4223
12. Getz NH, Marsden JE (1995) Joint-space tracking of workspace trajectories in continuous time. In: Proceedings of 34th IEEE conference on decision and control, pp 1001–1006
13. Getz NH, Marsden JE (1997) Dynamical methods for polar decomposition and inversion of matrices. Linear Algorithm Appl 258:311–343
14. Zhang Y, Yang Y, Tan N, Cai B (2011) Zhang neural network solving for time-varying full-rank matrix Moore-Penrose inverse. Computing 92(2):97–121
15. Zhang Y, Yi C (2011) Zhang neural networks and neural-dynamic method. Nova Science Publishers, New York
16. Zhang Y, Xie Y, Tan H (2012) Time-varying Moore-Penrose inverse solving shows different Zhang functions leading to different ZNN models. Lect Notes Comput Sci 7367:98–105

Chapter 10
Time-Varying Matrix Square Root

Abstract In this chapter, different indefinite ZFs, which lead to different ZD models, are proposed and developed as the error-monitoring functions for time-varying matrix square root finding. Toward the final purpose of field programmable gate array (FPGA) and application-specific integrated circuit (ASIC) realizations, the MATLAB Simulink modeling and verifications of such ZD models are further investigated to solve for time-varying matrix square root. Both theoretical analysis and modeling results substantiate the efficacy of the proposed ZD models for time-varying matrix square root finding.

10.1 Introduction

The problem of solving for matrix square root is considered to be an important special case of nonlinear matrix equation problem, which widely arises in many scientific and engineering fields; e.g., control theory [1], optimization [2], and signal processing [3]. In general, the solution of matrix square root, which can usually be a fundamental part of many solutions, can be achieved via matrix equations solving. Thus, many numerical algorithms/methods have been presented and developed for online solution of matrix square roots [1–7]. However, it may not be efficient enough for most numerical algorithms due to their serial-processing nature performed on digital computers [2, 3]. For large-scale online or real-time applications, the minimal arithmetic operations of such numerical algorithms are usually proportional to the cube of the matrix dimension n, i.e., $O(n^3)$ operations [8]. To remedy the inherent weaknesses of such numerical algorithms, many parallel-processing computational methods, including various dynamic system approaches, have been developed and implemented on specific architectures [9–15]. Note that the aforementioned computational schemes are theoretically/intrinsically designed for solving time-invariant (or termed, static, constant) problems (e.g., time-invariant matrix square root finding) rather than time-varying ones. Thus, these schemes may be less accurate and effective enough, when they are exploited directly to solve for time-varying matrix square root [16–18].

Y. Zhang and D. Guo, *Zhang Functions and Various Models*,
DOI 10.1007/978-3-662-47334-4_10

In this chapter, focusing on time-varying matrix square root finding, we propose, generalize, develop, and investigate eight different ZD models by defining eight different ZFs as the error-monitoring functions and constructing eight first-order differential equations to force the ZFs converge to zero. In addition to the theoretical analysis and results on the convergence characteristics of the proposed ZD models, the MATLAB Simulink modeling and verification examples are investigated with the final purpose of FPGA and ASIC realizations [19]. Moreover, some primary software modeling techniques are investigated to model and simulate such ZD models. The modeling results further substantiate the efficacy of the proposed ZD models based on different ZFs for time-varying square root finding.

10.2 ZFs and ZD Models

In this section, we introduce eight different ZFs and propose the resultant ZD models for time-varying matrix square root finding.

Let us consider the following time-varying matrix square root problem (which can also be viewed as a time-varying nonlinear matrix equation problem) [16–18]:

$$X^2(t) - A(t) = 0 \in \mathbb{R}^{n\times n}, \quad t \in [0, +\infty), \tag{10.1}$$

where $A(t) \in \mathbb{R}^{n\times n}$ denotes a smoothly time-varying positive-definite matrix, which, together with its time derivative $\dot{A}(t)$, is assumed to be known numerically or can be measured accurately. In addition, $X(t) \in \mathbb{R}^{n\times n}$ is the time-varying unknown matrix to be solved for. Our objective in this chapter is to find $X(t)$ so that (10.1) holds true for any $t \geqslant 0$. To lay a basis for further discussion, $A(t)$ is assumed to be nonsingular at any time instant $t \in [0, +\infty)$ in this chapter, and thus the inverse of $A(t)$ [(i.e., $A^{-1}(t)$] exists and is obtained.

Besides, the following preliminaries [1, 3, 18] are provided as a basis for further discussion on solving (10.1).

Definition 10.1 Given a smoothly time-varying matrix $A(t) \in \mathbb{R}^{n\times n}$, if matrix $X(t) \in \mathbb{R}^{n\times n}$ satisfies the time-varying nonlinear equation $X^2(t) = A(t)$, then $X(t)$ is a time-varying square root of matrix $A(t)$ [or say, $X(t)$ is a time-varying solution to the presented nonlinear equation (10.1)].

Definition 10.2 Since $X(t)X(t)X^{-1}(t)X^{-1}(t) = I$ (with $I \in \mathbb{R}^{n\times n}$ denoting the identity matrix) and $X(t)X(t) = X^2(t)$, then $X^{-2}(t)$ is defined as $X^{-2}(t) = X^{-1}(t)X^{-1}(t)$; i.e., we have $X^2(t)X^{-2}(t) = I$.

Concept 10.1 (Square-root existence condition) *If a smoothly time-varying matrix $A(t) \in \mathbb{R}^{n\times n}$ is positive-definite (in general sense [18]) at any time instant $t \in [0, +\infty)$, then there exists a time-varying matrix square root $X(t) \in \mathbb{R}^{n\times n}$ for matrix $A(t)$.*

Thus, specifically for solving time-varying matrix square root problem (10.1), in this chapter, we define eight different ZFs as follows:

$$E(t) = X^2(t)A^{-1}(t) - I, \tag{10.2}$$
$$E(t) = A^{-1}(t)X^2(t) - I, \tag{10.3}$$
$$E(t) = X^2(t) - A(t), \tag{10.4}$$
$$E(t) = X^{-2}(t) - A^{-1}(t), \tag{10.5}$$
$$E(t) = X(t) - A(t)X^{-1}(t), \tag{10.6}$$
$$E(t) = X(t) - X^{-1}(t)A(t), \tag{10.7}$$
$$E(t) = X^{-1}(t) - A^{-1}(t)X(t), \tag{10.8}$$
$$E(t) = X^{-1}(t) - X(t)A^{-1}(t). \tag{10.9}$$

Before deriving different ZD models from different ZFs, the following theorem is provided as a basis for further discussion.

Theorem 10.1 *The time derivative of $X^{-2}(t)$ [(i.e., $\mathrm{d}(X^{-2}(t))/\mathrm{d}t$] is formulated as*

$$\frac{\mathrm{d}(X^{-2}(t))}{\mathrm{d}t} = -X^{-2}(t)(X(t)\dot{X}(t) + \dot{X}(t)X(t))X^{-2}(t). \tag{10.10}$$

Proof It follows from Definition 10.2 that $X^2(t)X^{-2}(t) = I$. Then, we have

$$\frac{\mathrm{d}(X^2(t)X^{-2}(t))}{\mathrm{d}t} = \frac{\mathrm{d}I}{\mathrm{d}t} = 0.$$

Expanding the left-hand side of the above equation, we thus obtain

$$\frac{\mathrm{d}(X^2(t))}{\mathrm{d}t}X^{-2}(t) + X^2(t)\frac{\mathrm{d}(X^{-2}(t))}{\mathrm{d}t} = 0,$$

which is further rewritten as

$$X^2(t)\frac{\mathrm{d}(X^{-2}(t))}{\mathrm{d}t} = -\frac{\mathrm{d}(X^2(t))}{\mathrm{d}t}X^{-2}(t) = -(X(t)\dot{X}(t) + \dot{X}(t)X(t))X^{-2}(t).$$

Finally, in view of $X^2(t)X^{-2}(t) = I$, we have

$$\frac{\mathrm{d}(X^{-2}(t))}{\mathrm{d}t} = -X^{-2}(t)(X(t)\dot{X}(t) + \dot{X}(t)X(t))X^{-2}(t),$$

which now completes the proof. □

According to the ZD design formula (7.3), different ZFs lead to different ZD models for time-varying matrix square root finding, which is presented as follows.

Note that argument t [e.g., t in $X(t)$] is omitted in the following derivation of the ZD models for ease of presentation.

- Let us consider the ZD design formula (7.3), ZF (10.2), and Eq. (7.11) (see also Theorem 7.1). Then, we have

$$(\dot{X}X + X\dot{X})A^{-1} - X^2(A^{-1}\dot{A}A^{-1}) = -\gamma(X^2A^{-1} - I).$$

  Thus, based on ZF (10.2), we obtain the following dynamic equation (i.e., a first-order matrix-valued differential equation) of a ZD model for time-varying matrix square root finding:

$$\dot{X}X + X\dot{X} = X^2A^{-1}\dot{A} - \gamma(X^2 - A). \tag{10.11}$$

  In order to display ZD model (10.11) better, we can get its block diagram. Before doing this, we transform such a ZD model into the following explicit form:

$$\dot{X} = \dot{X}(I - X) - X\dot{X} + X^2A^{-1}\dot{A} - \gamma(X^2 - A).$$

  Therefore, we have the resultant block diagram of ZD model (10.11), which is shown in Fig. 10.1, and the modeling of ZD model (10.11) can also be done in this manner.
- Considering the ZD design formula (7.3), ZF (10.3), and Eq. (7.11), then we have

$$-A^{-1}\dot{A}A^{-1}X^2 + A^{-1}(\dot{X}X + X\dot{X}) = -\gamma(A^{-1}X^2 - I),$$

  which is reformulated as

$$\dot{X}X + X\dot{X} = \dot{A}A^{-1}X^2 - \gamma(X^2 - A). \tag{10.12}$$

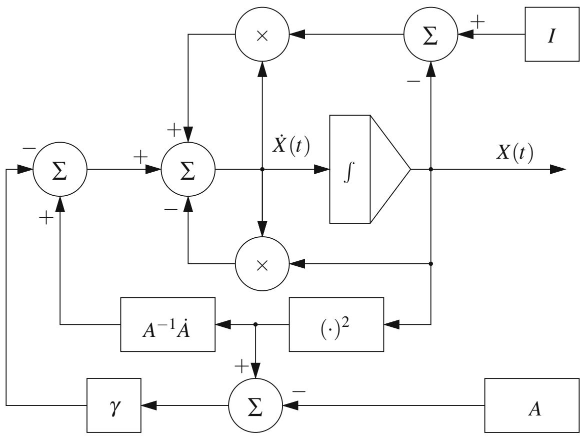


Fig. 10.1 Block diagram of ZD model (10.11) for time-varying matrix square root finding

That is, we obtain another ZD model (10.12), which is based on ZF (10.3), for time-varying matrix square root finding. In addition, the explicit form of ZD model (10.12) is

$$\dot{X} = (I - X)\dot{X} - \dot{X}X + \dot{A}A^{-1}X^2 - \gamma(X^2 - A).$$

It is worth pointing out that, by comparing the explicit form of the ZD model (10.11) with that of ZD model (10.12), we can see that the differences between these two explicit forms lie in the first three terms of the right-hand sides [for comparison, in the explicit form of (10.16), we have $\dot{X}(I - X)$, $X\dot{X}$, and $X^2A^{-1}\dot{A}$]. Due to the similarity, the block diagram of ZD model (10.12) is omitted and is left to interested readers to complete as a topic of exercise.

- With the ZD design formula (7.3) and ZF (10.4) exploited, the following ZD model is established for time-varying matrix square root finding:

$$X\dot{X} + \dot{X}X = -\gamma(X^2 - A) + \dot{A}. \tag{10.13}$$

Thus, ZD model (10.13) based on ZF (10.4) for time-varying matrix square root finding is obtained, and its explicit form is formulated as

$$\dot{X} = (I - X)\dot{X} - \dot{X}X + \dot{A} - \gamma(X^2 - A).$$

Note that, as compared to ZD models (10.11) and (10.12), ZD model (10.13) is viewed as a simplified one (i.e., with less model structure). The block diagram of ZD model (10.13) can thus be generalized from that of (10.11) or (10.12), and is omitted here due to the similarity (but is also left to interested readers to complete as a topic of exercise).

- With the ZD design formula (7.3), ZF (10.5), and Eqs. (7.11) and (10.10) exploited, we have

$$-X^{-2}(X\dot{X} + \dot{X}X)X^{-2} + A^{-1}\dot{A}A^{-1} = -\gamma(X^{-2} - A^{-1}),$$

which is reformulated as

$$X\dot{X} + \dot{X}X = X^2A^{-1}\dot{A}A^{-1}X^2 + \gamma(X^2 - X^2A^{-1}X^2). \tag{10.14}$$

Thus, we obtain another ZD model (10.14) based on ZF (10.5) for time-varying matrix square root finding. To depict the block diagram of ZD model (10.14), we transform such a ZD model into the following explicit form:

$$\dot{X} = \dot{X}(I - X) - X\dot{X} + X^2A^{-1}\dot{A}A^{-1}X^2 + \gamma(X^2 - X^2A^{-1}X^2).$$

Therefore, we have the resultant block diagram of ZD model (10.14) in Fig. 10.2.

- Considering the ZD design formula (7.3), ZF (10.6), and Eq. (7.10) (see also Theorem 7.1), we have

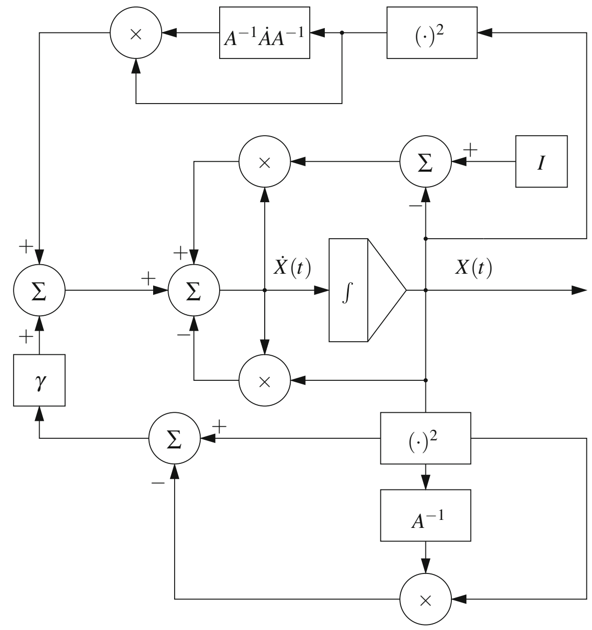


Fig. 10.2 Block diagram of ZD model (10.14) for time-varying matrix square root finding

$$\dot{X} - \dot{A}X^{-1} + AX^{-1}\dot{X}X^{-1} = -\gamma(X - AX^{-1}),$$

which is rewritten as

$$A^{-1}\dot{X} - A^{-1}\dot{A}X^{-1} + X^{-1}\dot{X}X^{-1} = -\gamma A^{-1}(X - AX^{-1}).$$

Then, we further have

$$XA^{-1}\dot{X}X - XA^{-1}\dot{A} + \dot{X} = -\gamma XA^{-1}(X - AX^{-1})X,$$

which is finally formulated as

$$\dot{X} = -XA^{-1}\dot{X}X + XA^{-1}\dot{A} - \gamma XA^{-1}(X^2 - A). \qquad (10.15)$$

Thus, based on ZF (10.6), ZD model (10.15) is obtained for time-varying matrix square root finding, of which the block diagram is shown in Fig. 10.3.

- Similar to the derivation of ZD model (10.15), based on ZF (10.7), we have

$$X\dot{X}A^{-1}X - \dot{A}A^{-1}X + \dot{X} = -\gamma X(X - X^{-1}A)A^{-1}X,$$

$$\dot{X} = -X\dot{X}A^{-1}X + \dot{A}A^{-1}X - \gamma(X^2 - A)A^{-1}X. \qquad (10.16)$$

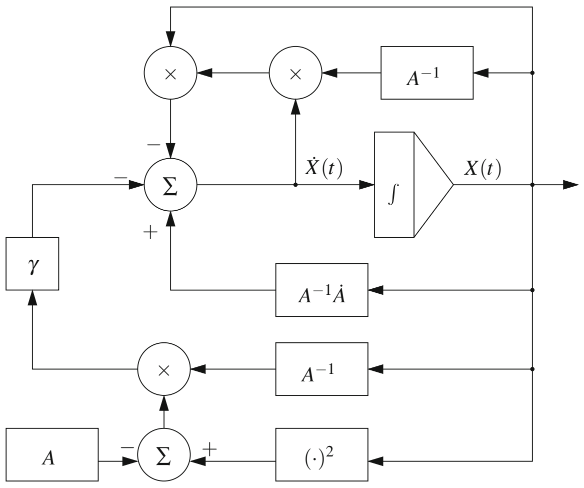


Fig. 10.3 Block diagram of ZD model (10.15) for time-varying matrix square root finding

Thus, we obtain ZD model (10.16) based on ZF (10.7) for time-varying matrix square root finding. Note that, like the situation of ZD model (10.11) and ZD model (10.12), the block diagram of ZD model (10.16) is omitted for its similarity to that of ZD model (10.15) (and is left to interested readers to complete as a topic of exercise).

- By using the ZD design formula (7.3), ZF (10.8), and Eqs. (7.10) and (7.11), we have

$$-X^{-1}\dot{X}X^{-1} + A^{-1}\dot{A}A^{-1}X - A^{-1}\dot{X} = -\gamma(X^{-1} - A^{-1}X),$$

and then

$$\dot{X} - XA^{-1}\dot{A}A^{-1}X^2 + XA^{-1}\dot{X}X = \gamma(X - XA^{-1}X^2),$$

which is finally formulated as

$$\dot{X} = XA^{-1}\dot{A}A^{-1}X^2 - XA^{-1}\dot{X}X - \gamma XA^{-1}(X^2 - A). \tag{10.17}$$

Therefore, based on ZF (10.8), ZD model (10.17) is obtained for time-varying matrix square root finding, of which the block diagram is shown in Fig. 10.4.

- Similar to the derivation of ZD model (10.17), based on ZF (10.9), we have

$$\dot{X} - X^2A^{-1}\dot{A}A^{-1}X + X\dot{X}A^{-1}X = \gamma(X - X^2A^{-1}X),$$

which is reformulated as

$$\dot{X} = X^2A^{-1}\dot{A}A^{-1}X - X\dot{X}A^{-1}X - \gamma(X^2 - A)A^{-1}X. \tag{10.18}$$

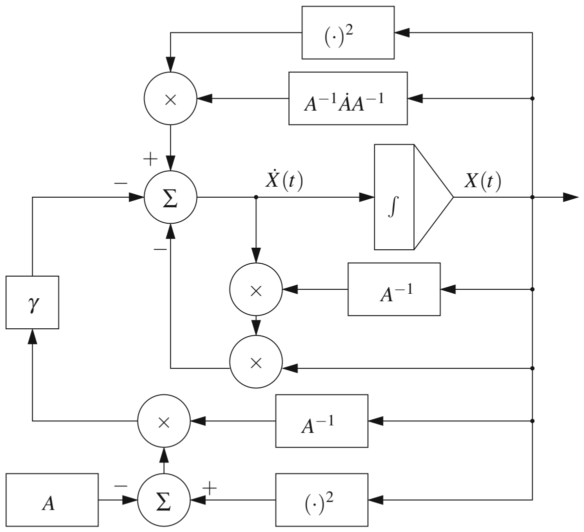


Fig. 10.4 Block diagram of ZD model (10.17) for time-varying matrix square root finding

Table 10.1 Different ZFs resulting in different ZD models for time-varying matrix square root finding

ZF	ZD model
(10.2)	$\dot{X}X + X\dot{X} = X^2A^{-1}\dot{A} - \gamma(X^2 - A)$
(10.3)	$\dot{X}X + X\dot{X} = \dot{A}A^{-1}X^2 - \gamma(X^2 - A)$
(10.4)	$X\dot{X} + \dot{X}X = \dot{A} - \gamma(X^2 - A)$
(10.5)	$X\dot{X} + \dot{X}X = X^2A^{-1}\dot{A}A^{-1}X^2 + \gamma(X^2 - X^2A^{-1}X^2)$
(10.6)	$\dot{X} = -XA^{-1}\dot{X}X + XA^{-1}\dot{A} - \gamma XA^{-1}(X^2 - A)$
(10.7)	$\dot{X} = -X\dot{X}A^{-1}X + \dot{A}A^{-1}X - \gamma(X^2 - A)A^{-1}X$
(10.8)	$\dot{X} = XA^{-1}\dot{A}A^{-1}X^2 - XA^{-1}\dot{X}X - \gamma XA^{-1}(X^2 - A)$
(10.9)	$\dot{X} = X^2A^{-1}\dot{A}A^{-1}X - X\dot{X}A^{-1}X - \gamma(X^2 - A)A^{-1}X$

Thus, we obtain ZD model (10.18) based on ZF (10.9) for time-varying matrix square root finding. Note that the block diagram of ZD model (10.18) is omitted for its similarity to that of ZD model (10.17).

In summary, we obtain eight different types of ZD models [i.e., ZD models (10.11)–(10.18)] for time-varying matrix square root finding, which correspond to eight different types of ZFs [i.e., ZFs (10.2)–(10.9)]. For readers' convenience and also for comparison, such eight different ZNN models corresponding to eight different ZFs are listed in Table 10.1.

10.3 Theoretical Results and Analyses

In this section, theoretical results and analyses are presented, which show the convergence performance of the proposed ZD models (10.11)–(10.18) on solving for time-varying matrix square root.

Theorem 10.2 *Consider a smoothly time-varying matrix $A(t) \in \mathbb{R}^{n\times n}$ involved in nonlinear equation (10.1), which satisfies the square-root existence condition. Starting from a randomly-generated positive-definite (or negative-definite) diagonal initial state-matrix $X(0) \in \mathbb{R}^{n\times n}$, the error function $E(t) = X^2(t)A^{-1}(t) - I \in \mathbb{R}^{n\times n}$ of ZD model (10.11), converges to zero [which implies that state matrix $X(t) \in \mathbb{R}^{n\times n}$ of ZD model (10.11) converges to the theoretical positive-definite (or negative-definite) time-varying matrix square root $X^*(t)$ of matrix $A(t)$].*

Proof From the compact form of the ZD design formula $\dot{E}(t) = -\gamma E(t)$, a set of n^2 decoupled differential equations can be written equivalently as follows:

$$\dot{e}_{ij}(t) = -\gamma e_{ij}(t), \tag{10.19}$$

for any $i \in \{1, 2, 3, \cdots, n\}$ and $j \in \{1, 2, 3, \cdots, n\}$. Thus, to analyze the equivalent ijth subsystem (10.19), we define a Lyapunov function candidate $v_{ij}(t) = e_{ij}^2(t)/2 \geqslant 0$ with its time derivative being

$$\frac{\mathrm{d}v_{ij}(t)}{\mathrm{d}t} = e_{ij}(t)\dot{e}_{ij}(t) = -\gamma e_{ij}^2(t) \leqslant 0,$$

which guarantees the final negative-definiteness of $\dot{v}_{ij}$ (i.e., $\dot{v}_{ij} < 0$ for $e_{ij} \neq 0$ while $\dot{v}_{ij} = 0$ for $e_{ij} = 0$ only). By Lyapunov theory [20, 21], the equilibrium point $e_{ij} = 0$ of (10.19) is asymptotically stable, i.e., $e_{ij}(t)$ converges to zero, for any $i \in \{1, 2, 3, \cdots, n\}$ and $j \in \{1, 2, 3, \cdots, n\}$. In other words, the matrix-valued error function $E(t) = [e_{ij}(t)] \in \mathbb{R}^{n\times n}$ is convergent to zero. In addition, we have $E(t) = X^2(t)A^{-1}(t) - I$, or equivalently, $X^2(t)A^{-1}(t) = I + E(t)$. Since $E(t) \to 0$ as $t \to +\infty$, we have $X^2(t)A^{-1}(t) \to I$ and thus $X^2(t) \to A(t)$ [(i.e., $X(t) \to X^*(t)$] as $t \to +\infty$. That is, the state matrix $X(t)$ of ZD model (10.11) can converge to the theoretical time-varying matrix square root $X^*(t)$ of matrix $A(t)$.

Furthermore, when the state matrix $X(t)$ of (10.11) starts from a randomly-generated positive-definite diagonal initial state-matrix $X(0)$, it can converge to the positive-definite time-varying matrix square root $A^{1/2}(t)$ [i.e., a form of $X^*(t)$]. This can be proven by the contradiction as follows. Suppose that the state matrix $X(t)$ starting from a positive-definite diagonal initial state-matrix $X(0)$ converges to the negative-definite time-varying matrix square root $-A^{1/2}(t)$ [i.e., the other form of $X^*(t)$], then such a state matrix $X(t)$ must pass through at least one 0-eigenvalue, which leads to the contradiction that the left- and right-hand sides of the ZD model (10.11) cannot hold. So, starting from a randomly-generated positive-definite diagonal initial state-matrix $X(0)$, the state matrix $X(t)$ of ZD model (10.11) converges to the positive-definite time-varying matrix square root $A^{1/2}(t)$. Similarly, it can also be proved that, starting from a randomly-generated negative-definite diagonal initial state-matrix $X(0)$, the state matrix $X(t)$ of ZD model (10.11) converges to the negative-definite time-varying matrix square root $-A^{1/2}(t)$ [i.e., another form of $X^*(t)$]. The proof is thus complete. □

As for the other seven ZD models (10.12)–(10.18), we also have the following convergence results, with the related proofs being generalized from the proof of Theorem 10.2 and being left to interested readers to complete as a topic of exercise.

Corollary 10.1 *Consider a smoothly time-varying matrix $A(t) \in \mathbb{R}^{n\times n}$ involved in nonlinear equation (10.1), which satisfies the square-root existence condition. Starting from a randomly-generated positive-definite (or negative-definite) diagonal initial state-matrix $X(0) \in \mathbb{R}^{n\times n}$, the error function $E(t) = A^{-1}(t)X^2(t) - I \in \mathbb{R}^{n\times n}$ of ZD model (10.12), converges to zero [which implies that the state matrix $X(t) \in \mathbb{R}^{n\times n}$ of ZD model (10.12) converges to theoretical positive-definite (or negative-definite) time-varying matrix square root $X^*(t)$ of matrix $A(t)$].*

Corollary 10.2 *Consider a smoothly time-varying matrix $A(t) \in \mathbb{R}^{n\times n}$ involved in nonlinear equation (10.1), which satisfies the square-root existence condition. Starting from a randomly-generated positive-definite (or negative-definite) diagonal initial state-matrix $X(0) \in \mathbb{R}^{n\times n}$, the error function $E(t) = X^2(t) - A(t) \in \mathbb{R}^{n\times n}$ of ZD model (10.13), converges to zero [which implies that the state matrix $X(t) \in \mathbb{R}^{n\times n}$ of ZD model (10.13) converges to theoretical positive-definite (or negative-definite) time-varying matrix square root $X^*(t)$ of matrix $A(t)$].*

Corollary 10.3 *Consider a smoothly time-varying matrix $A(t) \in \mathbb{R}^{n\times n}$ involved in nonlinear equation (10.1), which satisfies the square-root existence condition. Starting from a randomly-generated positive-definite (or negative-definite) diagonal initial state-matrix $X(0) \in \mathbb{R}^{n\times n}$, the error function $E(t) = X^{-2}(t) - A^{-1}(t) \in \mathbb{R}^{n\times n}$ of ZD model (10.14), converges to zero [which implies that the state matrix $X(t) \in \mathbb{R}^{n\times n}$ of ZD model (10.14) converges to theoretical positive-definite (or negative-definite) time-varying matrix square root $X^*(t)$ of matrix $A(t)$].*

Corollary 10.4 *Consider a smoothly time-varying matrix $A(t) \in \mathbb{R}^{n\times n}$ involved in nonlinear equation (10.1), which satisfies the square-root existence condition. Starting from a randomly-generated positive-definite (or negative-definite) diagonal initial state-matrix $X(0) \in \mathbb{R}^{n\times n}$, the error function $E(t) = X(t) - A(t)X^{-1}(t) \in \mathbb{R}^{n\times n}$ of ZD model (10.15), converges to zero [which implies that the state matrix $X(t) \in \mathbb{R}^{n\times n}$ of ZD model (10.15) converges to theoretical positive-definite (or negative-definite) time-varying matrix square root $X^*(t)$ of matrix $A(t)$].*

Corollary 10.5 *Consider a smoothly time-varying matrix $A(t) \in \mathbb{R}^{n\times n}$ involved in nonlinear equation (10.1), which satisfies the square-root existence condition. Starting from a randomly-generated positive-definite (or negative-definite) diagonal initial state-matrix $X(0) \in \mathbb{R}^{n\times n}$, the error function $E(t) = X(t) - X^{-1}(t)A(t) \in$*

$\mathbb{R}^{n\times n}$ of ZD model (10.16), converges to zero [which implies that the state matrix $X(t) \in \mathbb{R}^{n\times n}$ of ZD model (10.16) converges to theoretical positive-definite (or negative-definite) time-varying matrix square root $X^*(t)$ of matrix $A(t)$].

Corollary 10.6 *Consider a smoothly time-varying matrix $A(t) \in \mathbb{R}^{n\times n}$ involved in nonlinear equation (10.1), which satisfies the square-root existence condition. Starting from a randomly-generated positive-definite (or negative-definite) diagonal initial state-matrix $X(0) \in \mathbb{R}^{n\times n}$, the error function $E(t) = X^{-1}(t) - A^{-1}(t)X(t) \in \mathbb{R}^{n\times n}$ of ZD model (10.17), converges to zero [which implies that the state matrix $X(t) \in \mathbb{R}^{n\times n}$ of ZD model (10.17) converges to theoretical positive-definite (or negative-definite) time-varying matrix square root $X^*(t)$ of matrix $A(t)$].*

Corollary 10.7 *Consider a smoothly time-varying matrix $A(t) \in \mathbb{R}^{n\times n}$ involved in nonlinear equation (10.1), which satisfies the square-root existence condition. Starting from a randomly-generated positive-definite (or negative-definite) diagonal initial state-matrix $X(0) \in \mathbb{R}^{n\times n}$, the error function $E(t) = X^{-1}(t) - X(t)A^{-1}(t) \in \mathbb{R}^{n\times n}$ of ZD model (10.18), converges to zero [which implies that the state matrix $X(t) \in \mathbb{R}^{n\times n}$ of ZD model (10.18) converges to theoretical positive-definite (or negative-definite) time-varying matrix square root $X^*(t)$ of matrix $A(t)$].*

10.4 MATLAB Simulink Modeling

According to the aforementioned explicit forms and the presented block diagrams of the ZD models (10.11), (10.14), (10.15), and (10.17) shown in Figs. 10.1, 10.2, 10.3, and 10.4, the corresponding MATLAB Simulink modeling of such ZD models [i.e., ZD models (10.11), (10.14), (10.15), and (10.17)] is investigated and presented in this section for possible circuits implementation and also for the final purpose of FPGA and ASIC realizations.

10.4.1 Simulink Blocks

MATLAB Simulink contains a comprehensive block library including sinks, sources, linear, and nonlinear components, as well as connectors. The blocks generally used to construct ZD models (10.11), (10.14), (10.15), and (10.17) are discussed as follows.

- The *MATLAB Fcn* block can be employed to (1) generate matrix $A(t)$ using the *Clock* block as its input, or (2) compute the matrix norm.
- The *Constant* block, which outputs a constant specified by its parameter "Constant value", can be used to generate the identity matrix.
- The *Gain* block can be used to scale the neural network convergence, e.g., as a scaling parameter γ to scale the convergence rate of neural dynamics.

- The *Math Function* block can perform various common mathematical operations, and is used in this chapter for generating the inverse of a matrix.
- The *Product* block, specified as the standard matrix-wise product mode, can be used to multiply the matrices involved in the neural-dynamics models.
- The *Integrator* block makes continuous-time integration on the input signals. For instance, in the Example 10.1 discussed in the ensuing section, we set its "Initial condition" as "$diag(2 \ * \ rand(3,1))$" in order to generate a diagonal positive-definite initial state-matrix $X(0)$ with its diagonal elements randomly distributed in $[0, 2]$.

By interconnecting these basic Simulink function blocks and setting appropriate block parameters, the overall modeling of ZD models (10.11), (10.14), (10.15), and (10.17) can then be built up readily for time-varying matrix square roots finding, with the corresponding Simulink models shown in Figs. 10.5, 10.6, 10.7, and 10.8.

10.4.2 Parameter Settings

After showing the overall Simulink models of the proposed ZD models in Figs. 10.5, 10.6, 10.7 and 10.8, we discuss changing some of the default modeling environment options. The options setting can be done by using the "Configuration

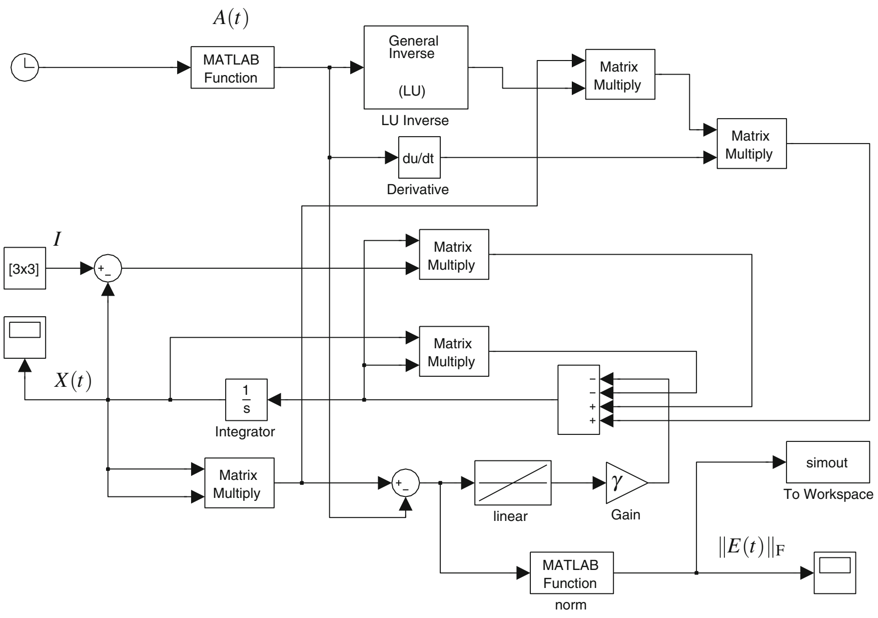


Fig. 10.5 Simulink modeling of ZD model (10.11) for time-varying matrix square root finding

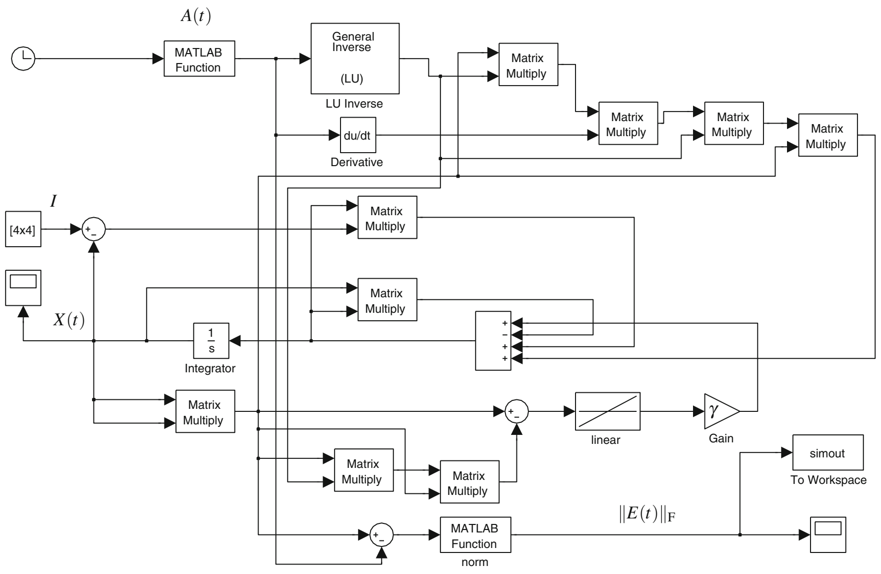


Fig. 10.6 Simulink modeling of ZD model (10.14) for time-varying matrix square root finding

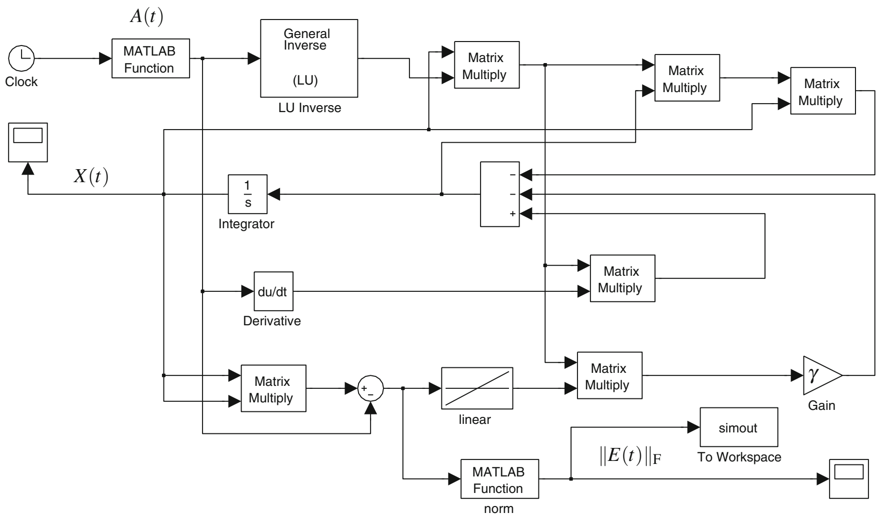


Fig. 10.7 Simulink modeling of ZD model (10.15) for time-varying matrix square root finding

Parameters" dialog box in the MATLAB Simulink environment [18]. Some important parameter settings have to be specified as follows:

- Starting time (e.g., 0.0) and Stop time (e.g., 8.0);
- Solver (i.e., integrator algorithm): "ode45 (Dormand-Prince)";

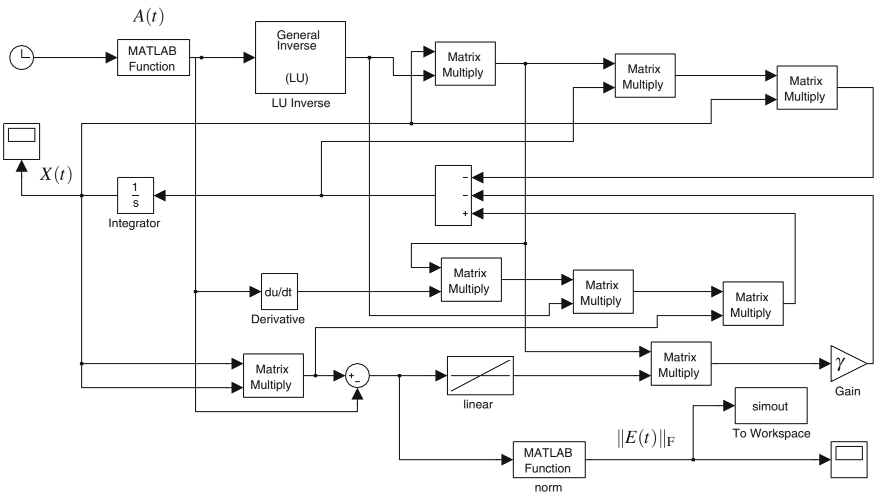


Fig. 10.8 Simulink modeling of ZD model (10.17) for time-varying matrix square root finding

- Max step size: “0.2”, and Min step size: “auto”;
- Initial step size: “auto”;
- Relative tolerance: “1e-6” (i.e., 10^{-6});
- Absolute tolerance:“auto”.

In addition, the check box in front of “States” of the option “Data Import/Export” should be selected, which is for the purpose of better displaying the ZD modeling results and is associated with the “StopFcn” code (of “Callbacks” in the dialog box entitled “Model Properties” which is started from the “File” pull-down menu).

10.5 Illustrative Examples

In the previous sections, different ZD models based on different ZFs have been proposed and developed for time-varying matrix square root finding, together with corresponding theoretical results. Based on the above-presented overall Simulink models depicted in Figs. 10.5, 10.6, 10.7, and 10.8, the ensuing illustrative examples are investigated to substantiate the efficacy of the proposed ZD models. Note that the representative ZD models (10.11), (10.14), (10.15), and (10.17) are chosen and modeled to solve for time-varying matrix square root.

Example 10.1 Let us consider nonlinear equation (10.1) with the following symmetric positive-definite time-varying matrix $A(t) \in \mathbb{R}^{3\times 3}$:

$$A(t) = \begin{bmatrix} 5+\sin^2(t) & 4\sin(t)+\exp(-2t) & 4+\exp(-2t)\sin(t) \\ 4\sin(t)+\exp(-2t) & 4+\sin^2(t)+\exp(-4t) & \sin(t)+4\exp(-2t) \\ 4+\exp(-2t)\sin(t) & \sin(t)+4\exp(-2t) & 5+\exp(-4t) \end{bmatrix}. \tag{10.20}$$

For such a matrix, the theoretical time-varying square root $X^*(t)$ is

$$X^*(t) = \begin{bmatrix} 2 & \sin(t) & 1 \\ \sin(t) & 2 & \exp(-2t) \\ 1 & \exp(-2t) & 2 \end{bmatrix} \in \mathbb{R}^{3\times3},$$

which is given for comparison purposes, i.e., to check the correctness of the neural dynamics solutions.

The proposed ZD models (10.11), (10.14), (10.15), and (10.17) are exploited to solve this problem, and the corresponding modeling results based on the above Simulink models are illustrated in Figs. 10.9, 10.10, 10.11, and 10.12. As shown in the left graph of Fig. 10.9, with design parameter $\gamma = 10$, the state matrix $X(t)$ of the proposed ZD model (10.11) denoted by solid curves converges to the theoretical time-varying solution $X^*(t)$ denoted by dash-dotted curves. In addition, to further investigate the convergence performance of ZD model (10.11), we monitor the residual error $\|E(t)\|_\mathrm{F} = \|X^2(t) - A(t)\|_\mathrm{F}$ during the solving process. As seen from the right graph of Fig. 10.9, by applying ZD model (10.11) to solve for time-varying matrix square root, the residual error converges to zero within around 1 s. For other ZD models [i.e., (10.14), (10.15), and (10.17)], we have the same observations/conclusions, which are shown in Figs. 10.10 and 10.11 (as well as the related modeling results which are omitted due to the similarity).

In addition, it is worth pointing out that the convergence performance of the proposed ZD models can be improved by increasing the value of γ. As an illustrative example, the convergence of residual error $\|E(t)\|_\mathrm{F}$ of ZD model (10.11) with different γ values is shown in Fig. 10.12. As seen from the figure, the convergence time of ZD model (10.11) can be expedited from around 8 s to 0.08 s and to 0.008 s, as the γ value is increased from 1 to 100 and to 1000, respectively. This result shows that ZD model (10.11) has an exponential convergence property, which can be expedited effectively by increasing the value of γ. Note that, for other ZD models [i.e., (10.14), (10.15), and (10.17)], we have the same conclusions by observing the related modeling results, which are omitted here due to the results' similarity. Being a topic of exercise, the corresponding modeling verifications of ZD models (10.14), (10.15), and (10.17) are left for interested readers.

In summary, the above modeling results (i.e., Figs. 10.9, 10.10, 10.11, and 10.12) have substantiated well the efficacy of the proposed ZD models (10.11), (10.14), (10.15), and (10.17) for time-varying matrix square root finding.

Example 10.2 In order to further investigate the efficacy of the proposed ZD models for larger dimension matrices, let us consider nonlinear equation (10.1) with the following time-varying circulant matrix $A(t)$:

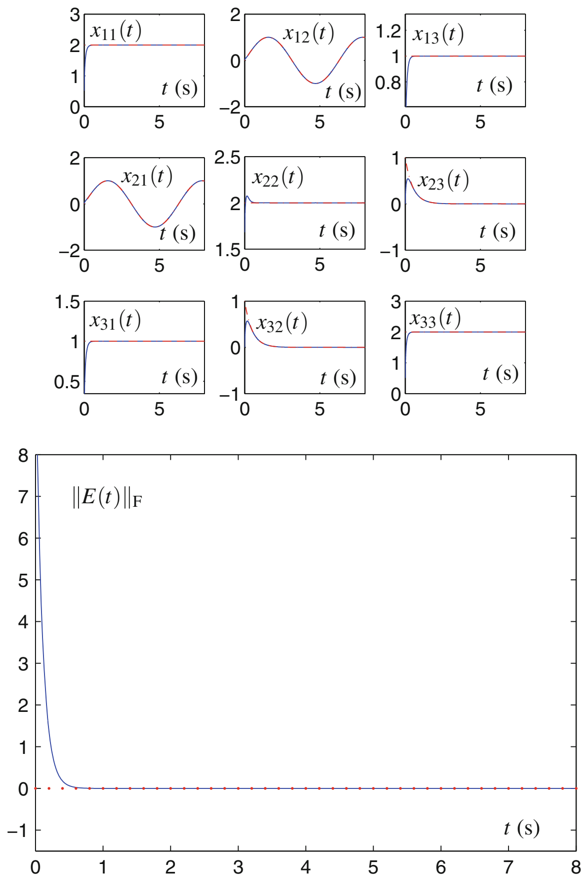


Fig. 10.9 Convergence performance of ZD model (10.11) with $\gamma = 10$ for finding the square root of time-varying matrix $A(t)$ in (10.20)

$$A(t) = \begin{bmatrix} a_0(t) & a_1(t) & a_2(t) & \cdots & a_{n-1}(t) \\ a_{n-1}(t) & a_0(t) & a_1(t) & \cdots & a_{n-2}(t) \\ a_{n-2}(t) & a_{n-1}(t) & a_0(t) & \cdots & a_{n-3}(t) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_1(t) & a_2(t) & a_3(t) & \cdots & a_0(t) \end{bmatrix} \in \mathbb{R}^{n\times n}, \tag{10.21}$$

in which, without loss of generality, we choose $a_0(t) = n$ and $a_i(t) = \sin(it)/i$ for $i = 1, 2, \cdots, n-1$. In this example, we choose $n = 6$. Evidently, the circulant matrix $A(t)$ is strictly diagonally dominant for any time instant $t \geqslant 0$.

Figure 10.13 shows the modeling results by using the proposed ZD models (10.11), (10.14), (10.15), and (10.17) with $\gamma = 10$ to find the time-varying matrix square root of the above circulant matrix $A(t)$. As seen from the figure, residual errors $\|E(t)\|_{\mathrm{F}}$ of such ZD models all converge to zero, which implies that their corresponding state matrices always converge to the theoretical time-varying square root of $A(t)$. These

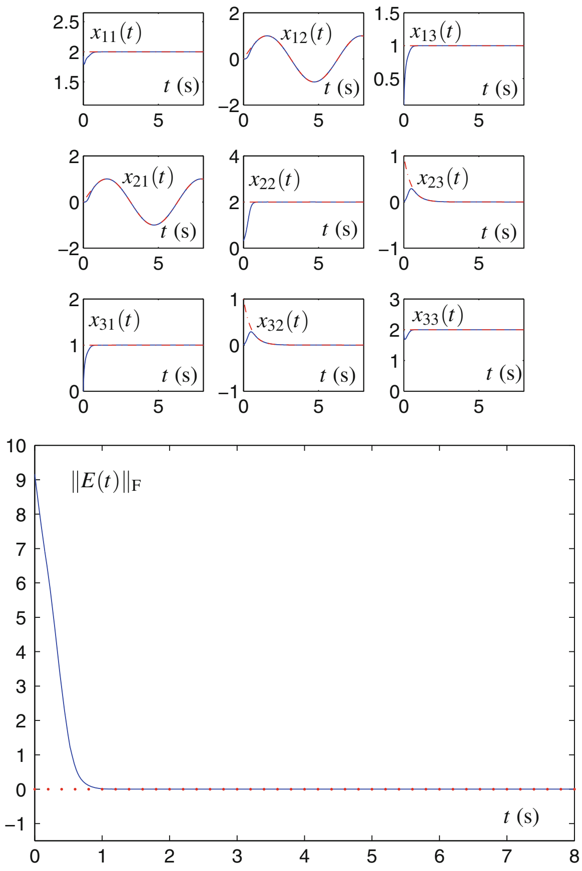


Fig. 10.10 Convergence performance of ZD model (10.14) with $\gamma = 10$ for finding the square root of time-varying matrix $A(t)$ in (10.20)

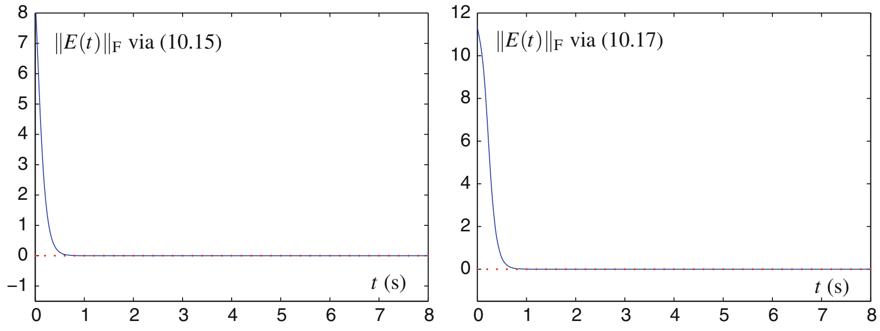


Fig. 10.11 Residual errors $\|E(t)\|_{\mathrm{F}}$ synthesized by ZD models (10.15) and (10.17) with $\gamma = 10$ for finding the square root of time-varying matrix $A(t)$ in (10.20)

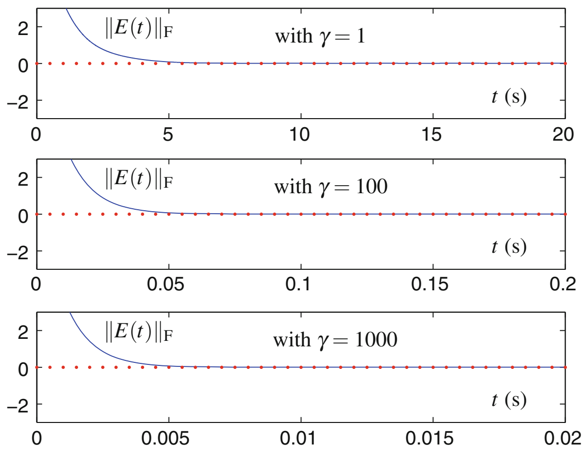


Fig. 10.12 Residual errors $\|E(t)\|_F$ synthesized by ZD model (10.11) with different values of γ (i.e., $\gamma = 1$, 100 and 1000) for finding the square root of time-varying matrix $A(t)$ in (10.20)

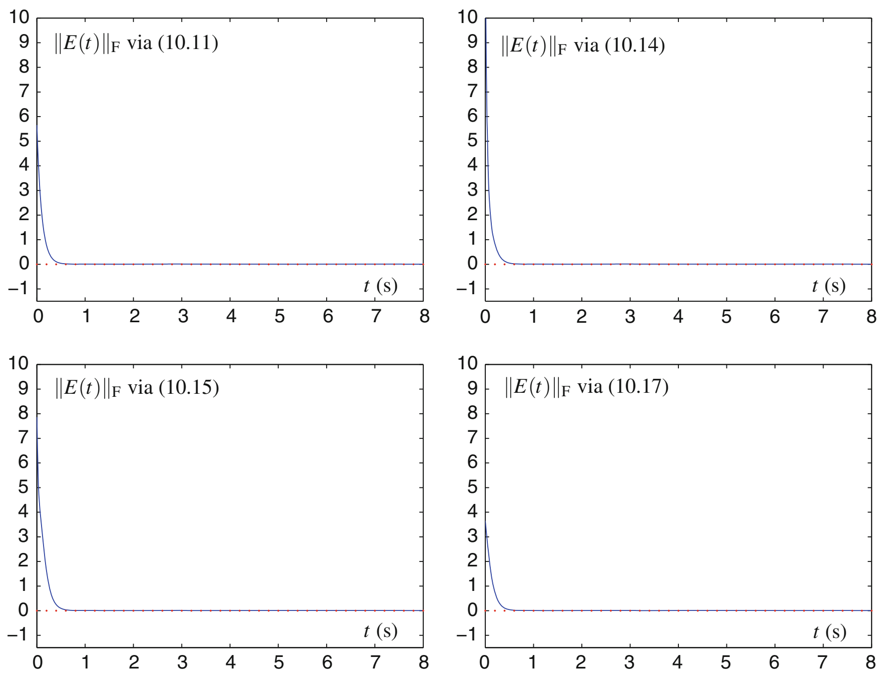


Fig. 10.13 Residual errors $\|E(t)\|_F$ synthesized by ZD models (10.11), (10.14), (10.15), and (10.17) with $\gamma = 10$ for finding the square root of time-varying circulant matrix $A(t)$ in (10.21)

results substantiate again the efficacy of the proposed ZD models (10.11), (10.14), (10.15), and (10.17) for time-varying matrix square root finding.

In summary, the above modeling results have shown the efficacy of the proposed ZD models (10.11), (10.14), (10.15), and (10.17) based on different ZFs for solving the time-varying matrix square root problem (10.1); and they have also confirmed the theoretical analysis and results given in Sect. 10.3. Besides, it is worth mentioning that the other ZD models [i.e., (10.12), (10.13), (10.16), and (10.18)] are also effectively exploited for time-varying matrix square root finding. The corresponding modeling verifications of such ZD models are left to interested readers to complete as a topic of exercise.

10.6 Summary

In this chapter, to solve for time-varying matrix square root, based on different ZFs (10.2)–(10.9), different ZD models shown in Table 10.1 [i.e., (10.11)–(10.18)] have been proposed, generalized, developed, and investigated in the form of the first-order matrix-valued differential equations. In addition, theoretical analysis and results have been given to show the convergence performance of such eight different ZD models. For possible hardware implementation based on electronic circuits, the MATLAB Simulink modeling of the proposed ZD models has been presented as well. Through illustrative computer-modeling examples, the efficacy of the proposed ZD models has been further substantiated for time-varying matrix square root finding [with the problem formulation depicted in (10.1)].

References

1. Higham NJ (1997) Stable iterations for the matrix square root. Numer Algorithms 15(2):227–242
2. Long JH, Hu XY, Zhang L (2007) Newton's method with exact line search for the square root of a matrix. In: Proceedings of international symposium on nonlinear dynamics, pp 1–5
3. Hasan MA, Hasan AA, Rahman S (2000) Fixed point iterations for computing square roots and the matrix sign function of complex matrices. In: Proceedings of 39th IEEE conference on decision and control, pp 4253–4258
4. Yao PF (1998) On the nonnegative square root of the fourth derivative operators. J Differ Equ 148(2):318–333
5. Johnson CR, Okubo K, Ream R (2001) Uniqueness of matrix square roots and an application. Linear Algebra Appl 323(1–3):51–60
6. Higham NJ (2002) Accuracy and stability of numerical algorithms, society for industrial and applied mathematics. SIAM, England
7. Higham NJ (2008) Functions of matrices: theory and computation, society for industrial and applied mathematics. SIAM, England

8. Zhang Y, Leithead WE, Leith DJ (2005) Time-series Gaussian process regression based on Toeplitz computation of $O(N^2)$ operations and $O(N)$-level storage. In: Proceedings of the 44th IEEE conference on decision and control and European control conference, pp 3711–3716
9. Babaei S, Geranmayeh A, Seyyedsalehi SA (2012) Towards designing modular recurrent neural networks in learning protein secondary structures. Expert Syst Appl 39(6):6263–6274
10. Zhang Y, Jiang D, Wang J (2002) A recurrent neural network for solving Sylvester equation with time-varying coefficients. IEEE Trans Neural Netw 13(5):1053–1063
11. Ghazali R, Hussain AJ, Liatsis P (2011) Dynamic ridge polynomial neural network: forecasting the univariate non-stationary and stationary trading signals. Expert Syst Appl 38(4):3765–3776
12. Zhang Y, Ge SS (2005) Design and analysis of a general recurrent neural network model for time-varying matrix inversion. IEEE Trans Neural Netw 16(6):1477–1490
13. Zhang Y, Ma W, Chen K, Li P (2007) MATLAB simulation of Zhang neural networks for time-varying Sylvester equation solving. In: Proceedings of the international conference on information computing and automation, pp 392–395
14. Wang J (1997) Recurrent neural networks for computing pseudoinverses of rank-deficient matrices. SIAM J Sci Comput 18(5):1479–1493
15. Übeyli ED (2010) Recurrent neural networks employing Lyapunov exponents for analysis of ECG signals. Expert Syst Appl 37(2):1192–1199
16. Zhang Y, Yang Y, Cai B, Guo D (2012) Zhang neural network and its application to Newton iteration for matrix square root estimation. Neural Comput Appl 21(3):453–460
17. Zhang Y, Yang Y (2008) Simulation and comparison of Zhang neural network and gradient neural network solving for time-varying matrix square roots. In: Proceedings of the 2nd international symposium on intelligent information technology application, pp 966–970
18. Zhang Y, Li W, Guo D, Ke Z (2013) Different Zhang functions leading to different ZNN models illustrated via time-varying matrix square roots finding. Expert Syst Appl 40(11):4393–4403
19. Shanblatt MA, Foulds B (2005) A Simulink-to-FPGA implementation tool for enhanced design flow. In: Proceedings of the IEEE international conference on microelectronic systems education, pp 89–90
20. Thieme HR (2011) Global stability of the endemic equilibrium in infinite dimension: Lyapunov functions and positive operators. J Differ Equ 250(9):3772–3801
21. Zhang Y, Yi C (2011) Zhang neural networks and neural-dynamic method. Nova Science Publishers, New York

Part IV
ZF in Complex Domain

Chapter 11
Time-Varying Complex Reciprocal

Abstract In Chap. 1, different ZD models based on different ZFs have been presented and investigated to solve for time-varying reciprocal in real domain. In this chapter, such a ZD approach (i.e., different ZFs leading to different ZD models) is extended and exploited for time-varying reciprocal computation in complex domain. Specifically, by defining four different ZFs, the corresponding four different ZD models are proposed, generalized, developed, and investigated to solve for time-varying complex reciprocal. Through three illustrative examples, the efficacy of the proposed complex ZD models for time-varying complex reciprocal finding is substantiated evidently.

11.1 Introduction

As presented in Chap. 1, the reciprocal computation described in the form of $f(x) = ax - 1 = 0$ is considered to be an important operation in a floating-point divider/processor. In the real world, we usually use the decimal arithmetic when presenting the numerical data and performing the computation. However, most of the microprocessors do not provide instructions or hardware support for the decimal floating-point arithmetic [1], and the digital computers can only process the binary numbers. As a result, the conversion between the decimal and binary numbers may introduce unacceptable errors. In addition, the high accuracy is often requested in the real-time applications.

For the floating-point divider, the Newton–Raphson iteration is an effective method with a quadratic convergence and can be faster than the digit recurrence methods if an accurate initial approximation is available [2]. Besides, there are also many other researches about the reciprocal computation, and thus various algorithms/methods (including the ZD models presented in Chap. 1) have been developed and studied for static and/or time-varying reciprocal finding [3–8]. Note that all of these algorithms/methods are just for reciprocal computation in real domain.

In recent years, the problems in the complex field have attracted the extensive attention of many researchers, and many relevant researches are arising in many science and engineering fields [9–13]. Under this background, in this chapter, instead of

Y. Zhang and D. Guo, *Zhang Functions and Various Models*,
DOI 10.1007/978-3-662-47334-4_11

solving time-varying reciprocal in real domain, the ZD approach (i.e., different ZFs leading to different ZD models) is extended and exploited for time-varying reciprocal computation in complex domain. That is, focusing on time-varying complex reciprocal finding, we propose, generalize, develop, and investigate four different ZD models by introducing four different ZFs. Note that the proposed complex ZD models based on the different complex ZFs are different from each other. Through three illustrative examples, the efficacy of the proposed complex ZD models for time-varying complex reciprocal finding is substantiated evidently.

11.2 Complex ZFs and ZD Models

In this section, we introduce four different complex ZFs (as the error-monitoring functions); and thus, the corresponding complex ZD models based on the presented complex ZFs are constructed to solve for time-varying complex reciprocal.

Without loss of generality, the time-varying complex reciprocal computation problem considered in this chapter is described in the following general form [9]:

$$f(z(t), t) = c(t)z(t) - 1 = 0 \in \mathbb{C},\ t \in [0, +\infty), \tag{11.1}$$

in which, $c(t) \neq 0 \in \mathbb{C}$ is a smoothly time-varying complex-valued scalar, with $\dot{c}(t)$ denoting the time derivative of $c(t)$. Note that, in this chapter, we suppose that $\dot{c}(t)$ is known or can be measured accurately in practice. Our objective is to find the unknown scalar $z(t) \in \mathbb{C}$ in real time $t \in [0, +\infty)$ such that (11.1) is always satisfied.

In view of that (11.1) is depicted in the scalar form, we reuse $e(t) \in \mathbb{C}$ and $\dot{e}(t) \in \mathbb{C}$ as the notations of the ZF and its time derivative in this chapter, respectively. Thus, the corresponding complex ZD design formula is formulated as follows [which is generalized from the ZD design formula (1.2) in real domain]:

$$\dot{e}(t) = \frac{\mathrm{d}e(t)}{\mathrm{d}t} = -\gamma e(t), \tag{11.2}$$

where design parameter $\gamma > 0 \in \mathbb{R}$ corresponds to the reciprocal of a capacitance parameter in the hardware implementation and should be set as large as the hardware would permit, e.g., in analog circuits or VLSI [14–16]. By exploiting the complex ZD design formula (11.2), the resultant complex ZD model can guarantee the global convergence performance in the process of solving time-varying complex reciprocal, since it utilizes the time-derivative information of the time-varying coefficient involved in (11.1). Note that design parameter γ plays an important role in the ZD approach, in the sense that different γ values can affect the convergence performance of the resultant complex ZD models.

To solve for time-varying complex reciprocal depicted in (11.1), we define the following four complex ZFs for constructing the complex ZD models:

$$e(t) = c(t)z(t) - 1 \in \mathbb{C}, \tag{11.3}$$

$$e(t) = z(t) - \frac{1}{c(t)} \in \mathbb{C}, \tag{11.4}$$

$$e(t) = \frac{1}{c(t)z(t)} - 1 \in \mathbb{C}, \tag{11.5}$$

$$e(t) = c(t) - \frac{1}{z(t)} \in \mathbb{C}. \tag{11.6}$$

According to the complex ZD design formula (11.2) and based on the complex ZFs (11.3)–(11.6), we can develop the corresponding complex ZD models for time-varying complex reciprocal finding as below.

- Let us consider complex ZF (11.3), of which the time derivative is

  $$\dot{e}(t) = \dot{c}(t)z(t) + c(t)\dot{z}(t).$$

  By exploiting the complex ZD design formula (11.2), we have

  $$\dot{c}(t)z(t) + c(t)\dot{z}(t) = -\gamma(c(t)z(t) - 1),$$

  which is reformulated as

  $$c(t)\dot{z}(t) = -\dot{c}(t)z(t) - \gamma(c(t)z(t) - 1). \tag{11.7}$$

  Thus, based on complex ZF (11.3), we obtain complex ZD model (11.7) for time-varying complex reciprocal finding. Through the same way, we can also exploit the other complex ZFs (11.4)–(11.6) to construct the other complex ZD models to solve for time-varying complex reciprocal.
- Considering complex ZF (11.4), we have its time derivative as

  $$\dot{e}(t) = \dot{z}(t) + \frac{1}{c^2(t)}\dot{c}(t).$$

  Together with the complex ZD design formula (11.2), we obtain

  $$\dot{z}(t) + \frac{1}{c^2(t)}\dot{c}(t) = -\gamma\left(z(t) - \frac{1}{c(t)}\right),$$

  which is reformulated as

  $$c^2(t)\dot{z}(t) = -\dot{c}(t) - \gamma(c^2(t)z(t) - c(t)). \tag{11.8}$$

Therefore, we have complex ZD model (11.8) for time-varying complex reciprocal finding, which is based on complex ZF (11.4).

- By combining complex ZF (11.5) and the complex ZD design formula (11.2), we have

$$-\frac{1}{c^2(t)z^2(t)}\left(\dot{c}(t)z(t)+c(t)\dot{z}(t)\right)=-\gamma\left(\frac{1}{c(t)z(t)}-1\right),$$

which is reformulated as

$$c(t)\dot{z}(t)=-\dot{c}(t)z(t)+\gamma(c(t)z(t)-c^2(t)z^2(t)). \tag{11.9}$$

As a result, complex ZD model (11.9) based on the corresponding complex ZF (11.5) is obtained for time-varying complex reciprocal finding.

- Similarly, by combining complex ZF (11.6) and the complex ZD design formula (11.2), we have

$$\dot{c}(t)+\frac{1}{z^2(t)}\dot{z}(t)=-\gamma\left(c(t)-\frac{1}{z(t)}\right),$$

which is reformulated as

$$\dot{z}(t)=-\dot{c}(t)z^2(t)-\gamma(c(t)z^2(t)-z(t)). \tag{11.10}$$

Therefore, we obtain complex ZD model (11.10) based on the corresponding complex ZF (11.6) for time-varying complex reciprocal finding.

In summary, we have obtained four different complex ZD models (11.7)–(11.10), which are, respectively, based on four different complex ZFs (11.3)–(11.6), for time-varying complex reciprocal finding. For readers' convenience and the purpose of summary, such complex ZFs and the corresponding complex ZD models are listed in Table 11.1. Evidently, from the proposed complex ZD models (11.7)–(11.10), we can find that, the complex-valued first-order time-derivative information of the time-varying coefficient involved in (11.1) is fully utilized. Hence, the global convergence performances of the proposed complex ZD models (11.7)–(11.10) are guaranteed.

Table 11.1 Different complex ZFs resulting in different complex ZD models for time-varying complex reciprocal finding

Complex ZF	Complex ZD model
(11.3)	$c(t)\dot{z}(t)=-\dot{c}(t)z(t)-\gamma(c(t)z(t)-1)$
(11.4)	$c^2(t)\dot{z}(t)=-\dot{c}(t)-\gamma(c^2(t)z(t)-c(t))$
(11.5)	$c(t)\dot{z}(t)=-\dot{c}(t)z(t)+\gamma(c(t)z(t)-c^2(t)z^2(t))$
(11.6)	$\dot{z}(t)=-\dot{c}(t)z^2(t)-\gamma(c(t)z^2(t)-z(t))$

Furthermore, the proposed complex ZD models (11.7)–(11.9) can be transferred from the implicit forms to the explicit forms as below.

- For complex ZD model (11.7), it is transferred as

$$\dot{z}(t) = (1 - c(t))\dot{z}(t) - \dot{c}(t)z(t) - \gamma(c(t)z(t) - 1).$$

- For complex ZD model (11.8), it is transferred as

$$\dot{z}(t) = \left(1 - c^2(t)\right)\dot{z}(t) - \dot{c}(t) - \gamma(c^2(t)z(t) - c(t)).$$

- For complex ZD model (11.9), it is transferred as

$$\dot{z}(t) = (1 - c(t))\dot{z}(t) - \dot{c}(t)z(t) + \gamma(c(t)z(t) - c^2(t)z^2(t)).$$

Thus, we have the block diagram of the proposed complex ZD model (11.7) depicted in Fig. 11.1 for the purpose of better understanding on its hardware implementation. Note that, as for other ZD models [i.e., (11.8)–(11.10)], similar block diagrams can be extended and obtained readily from Fig. 11.1. Being a topic of exercise, the corresponding block diagrams of such ZD models are left for interested readers.

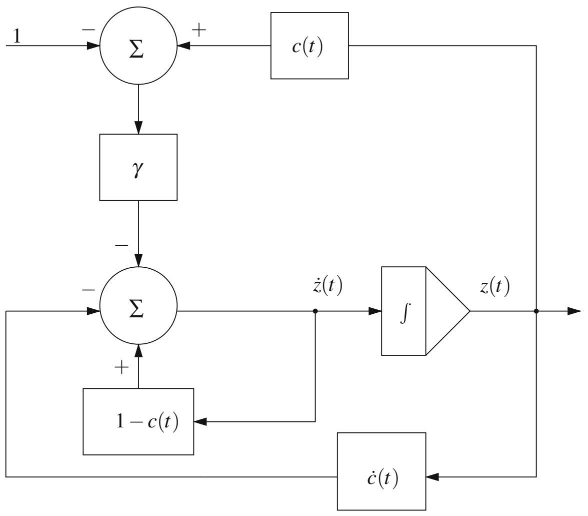


Fig. 11.1 Block diagram of complex ZD model (11.7) for time-varying complex reciprocal finding

11.3 Illustrative Examples

In the previous section, we have proposed and developed four different complex ZD models (11.7)–(11.10) based on different complex ZFs (11.3)–(11.6) for time-varying complex reciprocal finding. In this section, in order to substantiate the efficacy of the proposed complex ZD models (11.7)–(11.10), we have the following illustrative examples.

Example 11.1 In this example, let us consider the following time-varying complex reciprocal computation problem, in which the time-varying complex coefficient $c(t) = \cos(4t) + i\sin(5t)$ is involved, i.e.,

$$f(z(t), t) = (\cos(4t) + i\sin(5t))z(t) - 1 = 0 \in \mathbb{C}. \tag{11.11}$$

The proposed complex ZD models (11.7)–(11.10) with $\gamma = 10$ are exploited to solve the problem (11.11) and the corresponding simulation results are illustrated in Figs. 11.2, 11.3, 11.4 and 11.5. In each figure, the state trajectory of a complex ZD model for solving the time-varying complex reciprocal problem (11.11) is shown, as well as the trajectories of the real part and the imaginary part of the corresponding state. From the left graphs of Figs. 11.2, 11.3, 11.4 and 11.5, we can observe that, with $\gamma = 10$, the state trajectories of the proposed complex ZD models (11.7)–(11.10) are smooth and continuous, and can all converge to the theoretical time-varying solution of (11.11) rapidly and accurately. In addition, to make the above observation/conclusion more persuasive, as shown in the right graphs of Figs. 11.2, 11.3, 11.4 and 11.5, the trajectories of the real part and the imaginary part of the corresponding states of the proposed ZD models (11.7)–(11.10) can also converge to the real part and the imaginary part of the theoretical time-varying solution of (11.11), respectively. Thus, the efficacy of the proposed complex ZD models (11.7)–(11.10) is preliminarily substantiated.

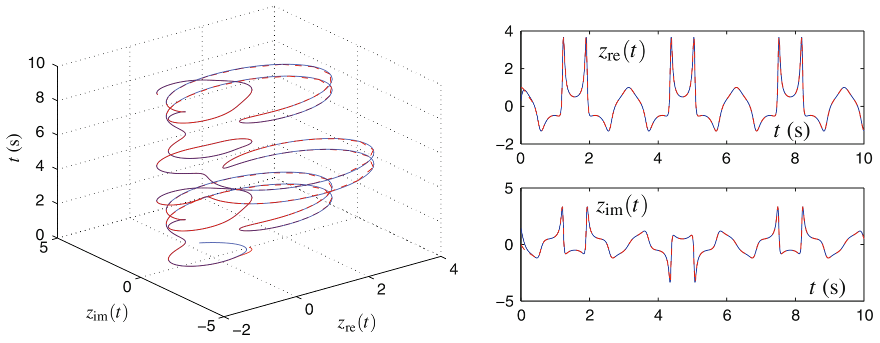


Fig. 11.2 State trajectories of complex ZD model (11.7) with $\gamma = 10$ for solving the time-varying complex reciprocal computation problem (11.11), where *dashed curves* correspond to the theoretical time-varying complex reciprocal, i.e., the theoretical solution of (11.11)

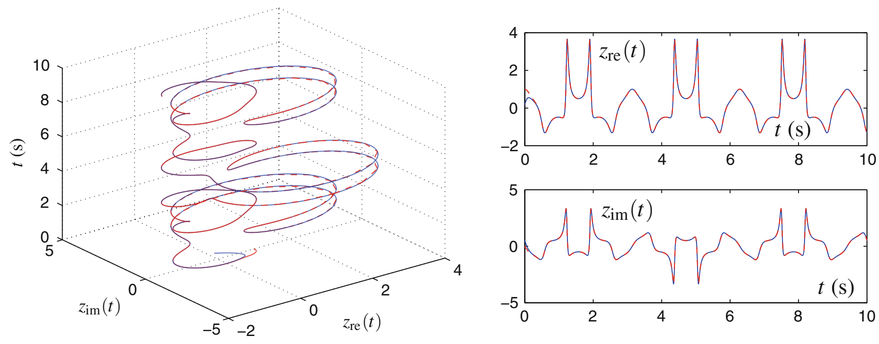


Fig. 11.3 State trajectories of the complex ZD model (11.8) with $\gamma = 10$ for solving the time-varying complex reciprocal computation problem (11.11)

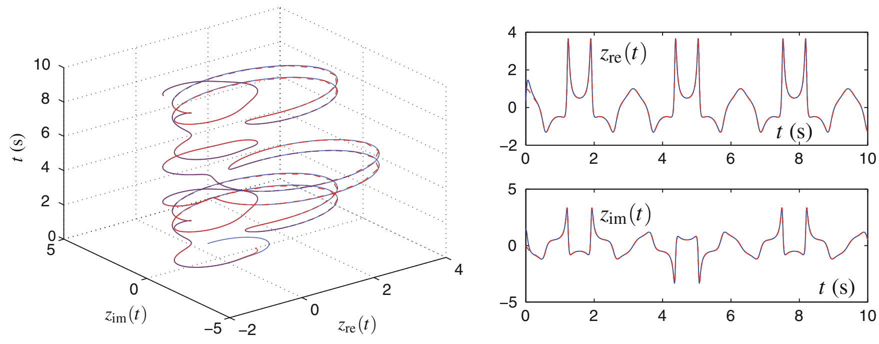


Fig. 11.4 State trajectories of the complex ZD model (11.9) with $\gamma = 10$ for solving the time-varying complex reciprocal computation problem (11.11)

Example 11.2 In the previous example, we have preliminarily substantiated the efficacy of the proposed complex ZD models (11.7)–(11.10). In this example, we use the residual error $e(t) = |c(t)z(t) - 1|$ (with symbol $|\cdot|$ denoting the absolute value of a scalar) to investigate the efficacy of the proposed complex ZD models (11.7)–(11.10) for time-varying complex reciprocal finding.

Let us consider the following time-varying complex reciprocal computation problem:

$$f(z(t), t) = (\cos(8t) + i \sin(4t))z(t) - 1 = 0 \in \mathbb{C}, \tag{11.12}$$

where the time-varying complex coefficient $c(t) = \cos(8t) + i \sin(4t)$ is used. Similarly, the proposed complex ZD models (11.7)–(11.10) are exploited to solve (11.12). The corresponding simulation results are displayed in Fig. 11.6. From Fig. 11.6, we can observe that, with $\gamma = 10$, the residual error $e(t)$ of each proposed complex ZD model can converge to 0 within a rather short time (i.e., about 0.76 s). In other words, these simulation results further demonstrate the efficacy of the proposed complex ZD models (11.7)–(11.10) for time-varying complex

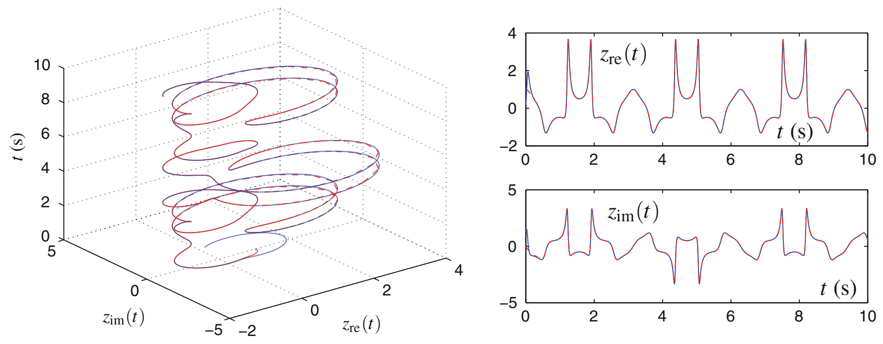


Fig. 11.5 State trajectories of the complex ZD model (11.10) with $\gamma = 10$ for solving the time-varying complex reciprocal computation problem (11.11)

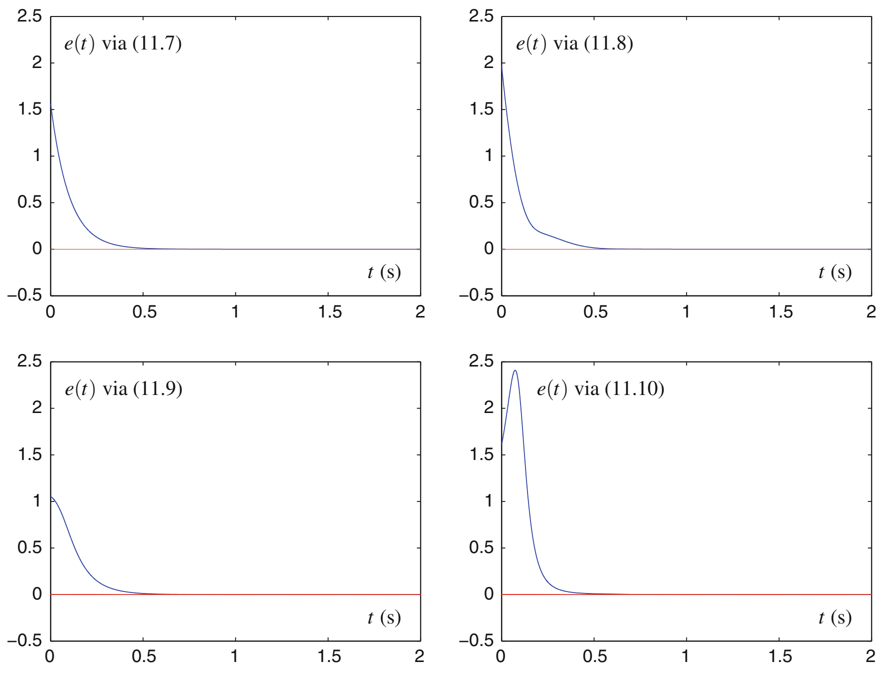


Fig. 11.6 Residual errors $e(t)$ synthesized by complex ZD models (11.7)–(11.10) with $\gamma = 10$ for solving the time-varying complex reciprocal computation problem (11.12)

reciprocal finding in the sense of the convergence performance of the residual error $e(t)$ for each complex ZD model.

Example 11.3 In this example, we investigate the important role of design parameter γ in the proposed complex ZD models (11.7)–(11.10). Let us consider the following time-varying complex reciprocal computation problem:

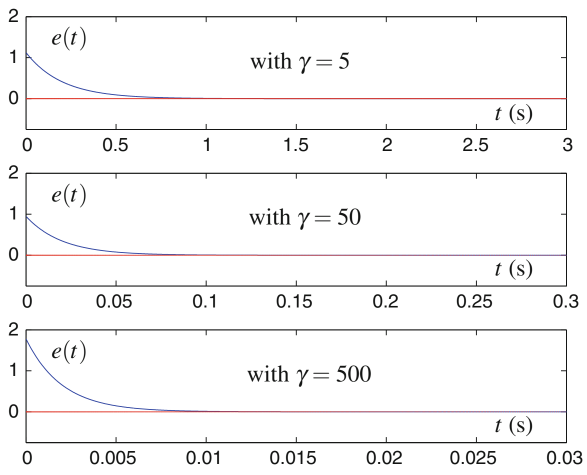


Fig. 11.7 Residual errors $e(t)$ of complex ZD model (11.7) with different γ values for solving the time-varying complex reciprocal computation problem (11.13)

$$f(z(t),t) = (\cos(5t) + i\sin(5t))z(t) - 1 = 0 \in \mathbb{C}. \qquad (11.13)$$

The corresponding simulation results are shown in Fig. 11.7 and Table 11.2 using the proposed complex ZD models (11.7)–(11.10) with different values of γ. In Fig. 11.7, an intuitive illustration is given for showing the change of the convergence time of the residual error $e(t)$ of complex ZD model (11.7). As seen from Fig. 11.7, the convergence time of the residual error $e(t)$ becomes much shorter when the value of design parameter γ increases; i.e., the convergence time of the residual error $e(t)$ decreases from about 1.41 s to about 0.015 s while the γ value increases from 5 to 500. The detailed convergence time of the proposed complex ZD models (11.7)–(11.10) with different γ values for solving the time-varying complex reciprocal computation problem (11.13) are shown in Table 11.2. From Fig. 11.7 and

Table 11.2 Convergence time (in seconds) of residual errors $e(t)$ of the proposed complex ZD models (11.7)–(11.10) using different values of γ for solving (11.13)

γ	ZD (11.7)	ZD (11.8)	ZD (11.9)	ZD (11.10)
1	7.40	7.40	6.95	7.00
2	3.68	3.70	3.53	3.70
5	1.41	1.40	1.40	1.42
10	0.76	0.74	0.72	0.75
20	0.38	0.37	0.36	0.39
50	0.14	0.14	0.15	0.14
100	0.070	0.076	0.072	0.070
200	0.035	0.038	0.037	0.036
500	0.015	0.015	0.014	0.015

Table 11.2, we can draw a conclusion that, different values of γ can affect the convergence performance of the proposed complex ZD models (11.7)–(11.10). That is, by increasing the value of γ, we can expedite the convergence of the residual errors $e(t)$ of the proposed complex ZD models (11.7)–(11.10).

In summary, through the above three illustrative examples, the efficacy of the proposed complex ZD models (11.7)–(11.10) based on complex ZFs (11.3)–(11.6) for solving time-varying complex reciprocal computation problem (11.1) has been evidently substantiated.

11.4 Summary

In this chapter, to solve for time-varying complex reciprocal [in the form of (11.1)], different complex ZD models [i.e., (11.7)–(11.10)] have been proposed, generalized, developed, and investigated by defining different complex ZFs [i.e., (11.3)–(11.6)]. The proposed complex ZD models fully utilize the complex-valued first-order time-derivative information of the time-varying coefficient involved in the time-varying complex reciprocal computation problem (11.1). Thus, the global convergence performance of the proposed complex ZD models is guaranteed. Through three illustrative examples, the efficacy of the proposed complex ZD models (11.7)–(11.10) has been evidently substantiated.

References

1. Wang L, Schulte MJ (2007) A decimal floating-point divider using Newton-Raphson iteration. J VLSI Signal Process Syst 49(1):3–18
2. Agrawal G, Khandelwal A, Swartzlander EE Jr (2007) An improved reciprocal approximation algorithm for a Newton-Raphson divider. In: Proceedings of advanced signal processing algorithms, architectures, and implementations, vol XVII, pp 1–12
3. Pineiro J, Bruguera JD (2002) High-speed double-precision computation of reciprocal, division, squareroot, and inverse square root. IEEE Trans Comput 51(12):1377–1388
4. Hanrot G, Rivat J, Tenenbaum G, Zimmermann P (2003) Density results on floating-point invertible numbers. Theor Comput Sci 291(2):135–141
5. Kucukkabak U, Akkas A (2004) Design and implementation of reciprocal unit using table look-up and Newton-Raphson iteration. In: Proceedings of euromicro symposium on digital system design, pp 249–253
6. Croot E, Li R-C, Zhu HJ (2004) The *abc* conjecture and correctly rounded reciprocal square roots. Theor Comput Sci 315(2–3):405–417
7. Antelo E, Lang T, Montuschi P, Sannarelli A (2005) Low latency digit-recurrence reciprocal and square-root reciprocal algorithm and architecture. In: Proceedings of the 17th IEEE symposium on computer arithmetic, pp 147–154
8. Zhang Y, Li F, Yang Y, Li Z (2012) Different Zhang functions leading to different Zhang-dynamics models illustrated via time-varying reciprocal solving. Appl Math Model 36(9):4502–4511

9. Zhang Y, Li Z, Guo D, Li F, Chen P (2012) Time-varying complex reciprocals solved by ZD via different complex Zhang functions. In: Proceedings of the 2nd international conference on computer science and network technology, pp 120–124
10. Song J, Yam Y (1998) Complex recurrent neural network for computing the inverse and pseudo-inverse of the complex matrix. Appl Math Comput 93(2–3):195–205
11. Yau L, Ben-Israel A (1998) The Newton and Halley methods for complex roots. Am Math Mon 105(9):806–818
12. Cardoso J, Loureiro A (2011) Iteration functions for *p*th roots of complex numbers. Numer Algorithms 57(3):329–356
13. Zhang Y, Li Z, Li K (2011) Complex-valued Zhang neural network for online complex-valued time-varying matrix inversion. Appl Math Comput 217(24):10066–10073
14. Mead C (1989) Analogue VLSI and neural systems. Addison-Wesley Longman, Boston
15. Zhang Y, Ma W, Li K, Yi C (2008) Brief history and prospect of coprocessors. China Sci Technol Inf 13:115–117
16. Zhang Y, Yi C (2011) Zhang neural networks and neural-dynamic method. Nova Science Publishers, New York

Chapter 12
Time-Varying Complex Matrix Inverse

Abstract In this chapter, the ZD approach is further extended and exploited for time-varying matrix inversion in complex domain. Specifically, focusing on time-varying complex matrix inversion, we propose, generalize, develop, and investigate three different complex ZD models by defining three different complex ZFs. Through simulations and verifications with four illustrative examples, the corresponding results substantiate the efficacy of the complex ZD models based on different complex ZFs for time-varying complex matrix inversion.

12.1 Introduction

As presented in Chap. 7, the problem of solving for constant matrix inverse in the form of $AX = I \in \mathbb{R}^{n\times n}$ and for time-varying matrix inverse in the form of $A(t)X(t) = I \in \mathbb{R}^{n\times n}$ is considered to be a fundamental issue and is widely encountered in science and engineering fields [1–9]. This is usually the essential part of many problems; for example, as the preliminary steps of image reconstruction [1], physics [2] and robot control [6, 8]. In many situations, the online solution of (time-varying) matrix inversion is required. As a result, a number of algorithms/methods (including those ZD models presented in Chap. 7) have been developed and investigated for constant and/or time-varying matrix inversion [7–15]. These research works, however, are confined to applying different numerical algorithms or neural dynamics to matrix inversion in real domain.

While it is true that more often the aforementioned problems involve only real-valued matrices, it is noted that in some situations complex-valued matrices may also occur, when the problem incorporates online frequency domain identification processes, or when the input signals contain both magnitude and phase information. The presence of a complex-valued matrix points to the need for efficient online complex matrix inversion as well [16–18]. Aiming at complex matrix inversion, Song and Yam [16] presented a complex neural network containing complex-valued weighting, thresholds, and inputs and output signals to yield the complex matrix inverse. Note that such a work is just for static (or termed, time-invariant) complex matrix inversion.

Therefore, in this chapter, based on the results presented in Chap. 7, the ZD approach (i.e., different ZFs leading to different ZD models) is further extended for time-varying complex matrix inversion (instead of static complex matrix inversion investigated in [16]). More specifically, by defining three different complex ZFs as the error-monitoring functions, three different complex ZD models are proposed, generalized, developed, and investigated to solve for time-varying complex matrix inverse. Some simulations and verifications are then performed, and from the illustrative results, the efficacy of the proposed complex ZD models based on different complex ZFs for time-varying complex matrix inversion is substantiated evidently.

12.2 Complex ZFs and ZD Models

Consider a smoothly time-varying complex-valued nonsingular matrix $C(t) \in \mathbb{C}^{n\times n}$. In this chapter, our objective is to find the unknown complex-valued matrix $Z(t) \in \mathbb{C}^{n\times n}$, which always satisfies the following time-varying complex matrix equation [17, 18]:

$$C(t)Z(t) = I \in \mathbb{C}^{n\times n},\ t \in [0, +\infty), \tag{12.1}$$

where I is the identity matrix, and the complex-valued matrix $C(t) \in \mathbb{C}^{n\times n}$ is usually assumed bounded in practice. In this chapter, we aim at solving the time-varying complex matrix-inversion problem (12.1) by constructing different complex ZD models based on different complex ZFs in an error-free manner (i.e., the residual error is small enough to be omitted). Note that, for the solvableness of the time-varying complex matrix-inversion problem (12.1), all of the eigenvalues of the complex-valued matrix $C(t) \in \mathbb{C}^{n\times n}$ must be nonzero.

As (12.1) is depicted in matrix form, we reuse $E(t) \in \mathbb{C}^{n\times n}$ and $\dot{E}(t) \in \mathbb{C}^{n\times n}$ as the notations of the ZF and its time derivative in this chapter, respectively. Thus, the corresponding complex ZD design formula is formulated as follows [which is generalized from the ZD design formula (7.3) in real domain]:

$$\dot{E}(t) = \frac{\mathrm{d}E(t)}{\mathrm{d}t} = -\gamma E(t), \tag{12.2}$$

where design parameter $\gamma > 0 \in \mathbb{R}$ is defined the same as before (see also Chap. 11). By exploiting the complex ZD design formula (12.2), the complex first-order time-derivative information of the time-varying complex matrix involved in (12.1), i.e., $\dot{C}(t) \in \mathbb{C}^{n\times n}$, is fully utilized in the process of solving for time-varying complex matrix inverse. Therefore, the resultant complex ZD model can guarantee the global convergence performance for time-varying complex matrix inversion.

In this chapter, specifically for time-varying complex matrix inversion with the problem formulation depicted in (12.1), we define the following complex ZFs as the error-monitoring functions:

$$E(t) = Z(t) - C^{-1}(t) \in \mathbb{C}^{n\times n}, \tag{12.3}$$
$$E(t) = C(t) - Z^{-1}(t) \in \mathbb{C}^{n\times n}, \tag{12.4}$$
$$E(t) = C(t)Z(t) - I \in \mathbb{C}^{n\times n}. \tag{12.5}$$

Before constructing the complex ZD models from such complex ZFs, we present the following theorem (i.e., an extension from real domain to complex domain) to lay a basis for further discussion.

Theorem 12.1 *The time derivative of* $C^{-1}(t)$, *i.e.,* $\mathrm{d}C^{-1}(t)/\mathrm{d}t$, *is formulated as* $\dot{C}^{-1}(t) = \mathrm{d}C^{-1}(t)/\mathrm{d}t = -C^{-1}(t)\dot{C}(t)C^{-1}(t)$. *So as for the time derivative of* $Z^{-1}(t)$, *i.e.,* $\dot{Z}^{-1}(t) = \mathrm{d}Z^{-1}(t)/\mathrm{d}t = -Z^{-1}(t)\dot{Z}(t)Z^{-1}(t)$.

Proof It is generalized from Theorem 7.1 in Chap. 7. □

According to the complex ZD design formula (12.2), by using different complex ZFs [i.e., (12.3)–(12.5)], three different complex ZD models for time-varying complex matrix inversion are thus derived and presented as follows.

- Based on Theorem 12.1, let us consider the complex ZD design formula (12.2) and complex ZF (12.3). Then, we have

$$\dot{Z}(t) - \dot{C}^{-1}(t) = -\gamma\left(Z(t) - C^{-1}(t)\right),$$

  which is further rewritten as

$$\begin{aligned}\dot{Z}(t) + C^{-1}(t)\dot{C}(t)C^{-1}(t) &= -\gamma\left(Z(t) - C^{-1}(t)\right),\\ \dot{Z}(t) &= -C^{-1}(t)\dot{C}(t)C^{-1}(t) - \gamma\left(Z(t) - C^{-1}(t)\right),\\ C(t)\dot{Z}(t)C(t) &= -\dot{C}(t) - \gamma\left(C(t)Z(t)C(t) - C(t)\right).\end{aligned} \tag{12.6}$$

  Thus, we obtain complex ZD model (12.6) for time-varying complex matrix inversion. For further illustration and investigation, we can also exploit other complex ZFs [i.e., complex ZFs (12.4) and (12.5)] to construct other types of complex ZD models.
- Based on Theorem 12.1, considering the complex ZD design formula (12.2) and complex ZF (12.4), we have the following derivation:

$$\begin{aligned}\dot{C}(t) - \dot{Z}^{-1}(t) &= -\gamma\left(C(t) - Z^{-1}(t)\right),\\ \dot{C}(t) + Z^{-1}(t)\dot{Z}(t)Z^{-1}(t) &= -\gamma\left(C(t) - Z^{-1}(t)\right),\end{aligned}$$

Table 12.1 Different complex ZFs resulting in different complex ZD models for time-varying complex matrix inversion

Complex ZF	Complex ZD model
(12.3)	$C(t)\dot{Z}(t)C(t) = -\dot{C}(t) - \gamma(C(t)Z(t)C(t) - C(t))$
(12.4)	$\dot{Z}(t) = -Z(t)\dot{C}(t)Z(t) - \gamma(Z(t)C(t)Z(t) - Z(t))$
(12.5)	$C(t)\dot{Z}(t) = -\dot{C}(t)Z(t) - \gamma(C(t)Z(t) - I)$

$$Z(t)\dot{C}(t)Z(t) + \dot{Z}(t) = -\gamma Z(t)\left(C(t) - Z^{-1}(t)\right)Z(t),$$
$$\dot{Z}(t) = -Z(t)\dot{C}(t)Z(t) - \gamma\left(Z(t)C(t)Z(t) - Z(t)\right). \tag{12.7}$$

Therefore, another type of complex ZD model, i.e., complex ZD model (12.7), is obtained for time-varying complex matrix inversion.

- By combining the complex ZD design formula (12.2) and complex ZF (12.5), we have
$$C(t)\dot{Z}(t) = -\dot{C}(t)Z(t) - \gamma\left(C(t)Z(t) - I\right). \tag{12.8}$$
Thus, complex ZD model (12.8) based on complex ZF (12.5) for time-varying complex matrix inversion is obtained.

In summary, we have constructed three different types of complex ZD models, i.e., complex ZD models (12.6)–(12.8), by defining three different complex ZFs [i.e., complex ZFs (12.3)–(12.5)] for solving the time-varying complex matrix-inversion problem (12.1). Besides, for readers' convenience and also for comparison, the above three different ZD models corresponding to three different ZFs are listed in Table 12.1. Note that, compared with complex ZD model (12.6), complex ZD model (12.8) has a relatively simplified model structure. Thus, in the ensuing simulations, such a ZD model (12.8) is not discussed [since it can be generalized from (12.6)]. Being a topic of exercise, the corresponding simulative verification of complex ZD model (12.8) is left for interested readers (see also [14]).

12.3 Illustrative Examples

In the previous sections, we developed complex ZD models (12.6) and (12.7). Note that both the proposed complex ZD models for time-varying complex matrix inversion are described in matrix form. Thus, vectorization techniques are needed to transform such matrix-form differential equations to vector-form differential equations for simulative purposes.

- For complex ZD model (12.6), based on the Kronecker product (denoted by the symbol of "⊗") and vectorization techniques, we can transform (12.6) into the following vector-form differential equation:

$$\begin{aligned}\left(C^{\mathrm{T}}(t)\otimes C(t)\right)\operatorname{vec}\left(\dot{Z}(t)\right) &= -\operatorname{vec}\left(\dot{C}(t)\right)\\ &\quad -\gamma\left(\left(C^{\mathrm{T}}(t)\otimes C(t)\right)\operatorname{vec}(Z(t))-\operatorname{vec}(C(t))\right).\end{aligned}$$

- Similarly, for complex ZD model (12.7), we can have its vector form as

$$\begin{aligned}\operatorname{vec}\left(\dot{Z}(t)\right) &= -\left(I\otimes\left(Z(t)\dot{C}(t)\right)\right)\operatorname{vec}(Z(t))\\ &\quad -\gamma\left(\left(I\otimes\left(Z(t)C(t)\right)\right)\operatorname{vec}(Z(t))-\operatorname{vec}(Z(t))\right).\end{aligned}$$

In this section, we have the ensuing examples for verification of the efficacy of the proposed complex ZD models (12.6) and (12.7).

Example 12.1 Let us consider the following time-varying complex-valued matrix $C(t)$ involved in (12.1):

$$C(t)=\begin{bmatrix}\exp(10it) & -i\exp(-10it)\\ -i\exp(10it) & \exp(-10it)\end{bmatrix}\in\mathbb{C}^{2\times 2}, \qquad (12.9)$$

whose inverse, i.e., the theoretical inverse of (12.9), is given as

$$C^{-1}(t)=\begin{bmatrix}0.5\exp(10it) & 0.5i\exp(10it)\\ 0.5i\exp(-10it) & 0.5\exp(-10it)\end{bmatrix}\in\mathbb{C}^{2\times 2}.$$

Since we have obtained the theoretical time-varying inverse of matrix $C(t)$, we can use it as an analytic theoretical solution to verify the correctness of the solutions of the proposed complex ZD models (12.6) and (12.7). As illustrated in Figs. 12.1 and 12.2, starting from the randomly-generated initial states $Z(0)\in\mathbb{C}^{2\times 2}$, the state matrices $Z(t)\in\mathbb{C}^{2\times 2}$ of complex ZD models (12.6) and (12.7) with design parameter $\gamma=10$ can converge to the theoretical time-varying inverse $C^{-1}(t)$ rapidly and accurately within a rather short time. Thus, from these results, the efficacy of the proposed complex ZD models (12.6) and (12.7) for time-varying complex matrix inversion is substantiated primarily.

Example 12.2 In this example, we will further verify the efficacy of the proposed complex ZD models (12.6) and (12.7). Let us consider the following time-varying complex-valued matrix $C(t)$ involved in (12.1):

$$C(t)=\begin{bmatrix}i\sin(5t) & i\cos(5t)\\ i\cos(5t) & -i\sin(5t)\end{bmatrix}\in\mathbb{C}^{2\times 2}, \qquad (12.10)$$

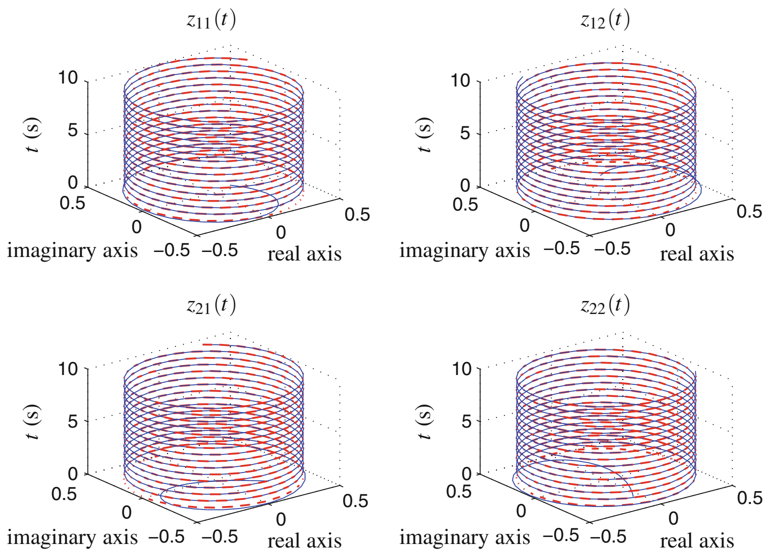


Fig. 12.1 State trajectories of complex ZD model (12.6) with $\gamma = 10$ for inverting the time-varying complex-valued matrix in (12.9), where *dash-dotted curves* denote the theoretical time-varying inverse and *solid curves* denote the solution computed by complex ZD model (12.6)

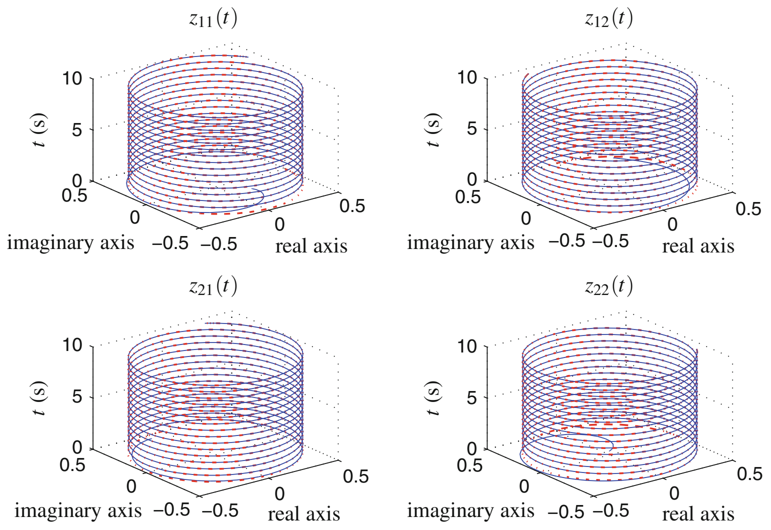


Fig. 12.2 State trajectories of complex ZD model (12.7) with $\gamma = 10$ for inverting the time-varying complex-valued matrix in (12.9), where *dash-dotted curves* denote the theoretical time-varying inverse and *solid curves* denote the solution computed by complex ZD model (12.7)

and its theoretical time-varying inverse is given as

$$C^{-1}(t) = \begin{bmatrix} -i\sin(5t) & -i\cos(5t) \\ -i\cos(5t) & i\sin(5t) \end{bmatrix} \in \mathbb{C}^{2\times 2}.$$

In this example, we investigate the residual errors $\|E(t)\|_{\mathrm{F}} = \|C(t)Z(t) - I\|_{\mathrm{F}}$ synthesized by the proposed complex ZD models (12.6) and (12.7) with $\gamma = 10$. From the residual errors illustrated in Fig. 12.3, we can find that the residual error of each of complex ZD models (12.6) and (12.7) diminishes to 0 within about 1 s. Therefore, the efficacy of the proposed complex ZD models (12.6) and (12.7) for time-varying complex matrix inversion is further substantiated.

Example 12.3 In this example, we consider a more complicated situation of the time-varying complex matrix-inversion problem (12.1), which is the inversion of the following time-varying complex-valued Toeplitz matrix: $C(t)$:

$$C(t) = \begin{bmatrix} c_1(t) & c_2(t) & c_3(t) & \cdots & c_n(t) \\ c_2(t) & c_1(t) & c_2(t) & \cdots & c_{n-1}(t) \\ c_3(t) & c_2(t) & c_1(t) & \cdots & c_{n-2}(t) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_n(t) & c_{n-1}(t) & c_{n-2}(t) & \cdots & c_1(t) \end{bmatrix} \in \mathbb{C}^{n\times n}. \tag{12.11}$$

Let $c_1(t) = n + \exp(10it)$ and $c_k(t) = \exp(-10it)/(k-1)(k = 2, 3, \ldots n)$. We know that the time-varying complex-valued Toeplitz matrix $C(t)$ is strictly diagonally dominant for any time instant $t \geqslant 0$ and is invertible. To verify the efficacy of the proposed complex ZD models (12.6) and (12.7) for a more complicated time-varying complex matrix-inversion problem solving, we exploited the complex ZD models (12.6) and (12.7) to solve for the inverse of $C(t)$ in (12.11) in the situation of $n = 6$. The corresponding simulation results are shown in Fig. 12.4. Evidently, the residual errors $\|E(t)\|_{\mathrm{F}}$ synthesized by complex ZD models (12.6) and (12.7) with $\gamma = 10$

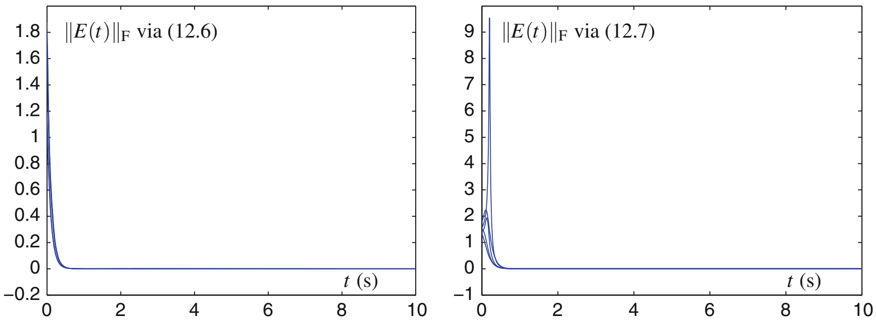


Fig. 12.3 Residual errors $\|E(t)\|_{\mathrm{F}} = \|C(t)Z(t) - I\|_{\mathrm{F}}$ synthesized by complex ZD models (12.6) and (12.7) with $\gamma = 10$ for the inversion of matrix $C(t)$ in (12.10)

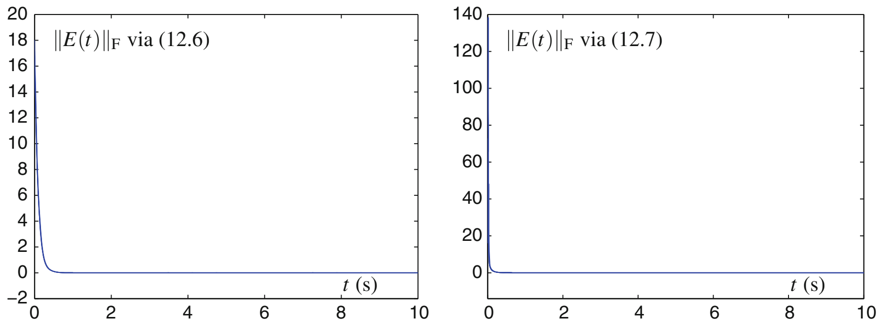
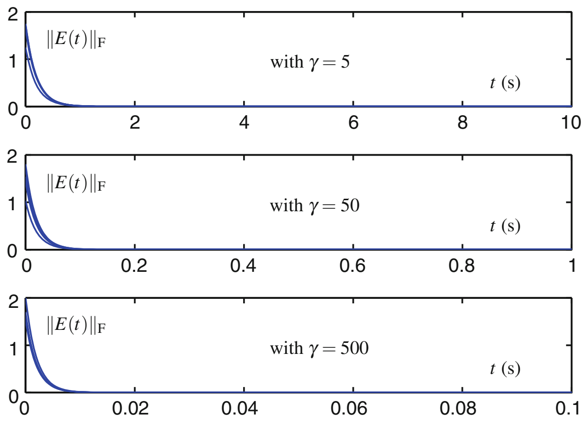


Fig. 12.4 Residual errors $\|E(t)\|_F = \|C(t)Z(t) - I\|_F$ synthesized by complex ZD models (12.6) and (12.7) with $\gamma = 10$ for the inversion of matrix $C(t)$ in (12.11)

can diminish to 0 within a short time, which means that the corresponding solutions $Z(t)$ converge to the theoretical time-varying inverse rapidly and accurately. As a result, the efficacy of the proposed complex ZD models (12.6) and (12.7) for a more complicated time-varying complex matrix-inversion problem solving is substantiated evidently.

Example 12.4 In this example, we investigate the important role of the design parameter γ. In order to achieve the faster convergence of the proposed complex ZD models (12.6) and (12.7) for time-varying complex matrix inversion with matrix $C(t)$ depicted in (12.10), we can increase the value of γ correspondingly. As displayed in Fig. 12.5, we can clearly find that the residual errors $\|E(t)\|_F$ synthesized by complex ZD model (12.6) diminish more rapidly with the increase of the γ values (i.e., with $\gamma = 5$, 50, and 500, respectively). That is, with $\gamma = 5$, 50, and 500, the convergence time of the residual error $\|E(t)\|_F$ diminishes from about 1.4 s to about 0.014 s. Note that the simulation results by using complex ZD model (12.7)

Fig. 12.5 Comparison of the residual errors $\|E(t)\|_F$ synthesized by complex ZD model (12.6) with $\gamma = 5$, 50, and 500 for the inversion of matrix $C(t)$ in (12.10)



are similar to those shown in Fig. 12.5 and are thus omitted due to results similarity (but with the corresponding verification being left to interested readers to complete as a topic of exercise). Therefore, we can draw the conclusion that, we can promote the convergence performance of the proposed complex ZD models (12.6) and (12.7) by choosing a larger value of design parameter γ.

In sum, from the above four illustrative examples, we have already substantiated the efficacy of the proposed complex ZD models (12.6) and (12.7) for the time-varying complex matrix inversion.

12.4 Summary

In this chapter, to solve the time-varying complex matrix-inversion problem (12.1), three different complex ZD models [i.e., (12.6)–(12.8)] have been proposed, generalized, developed, and investigated by defining different complex ZFs [i.e., (12.3)–(12.5)]. Such complex ZD models utilize the complex first-order time-derivative information of the time-varying complex matrix involved in (12.1) and achieve the global convergence performance. Through four illustrative examples, the efficacy of the proposed complex ZD models for time-varying complex matrix inversion has been substantiated evidently.

References

1. Steriti RJ, Fiddy MA (1993) Regularized image reconstruction using SVD and a neural network method for matrix inversion. IEEE Trans Signal Process 41(10):3074–3077
2. Van Huffel S, Vandewalle J (1989) Analysis and properties of the generalized total least squares problem $AX \approx B$ when some or all columns in A are subject to error. SIAM J Matrix Anal Appl 10(3):294–315
3. Chen M, Ge SS, How BVE (2010) Robust adaptive neural network control for a class of uncertain MIMO nonlinear systems with input nonlinearities. IEEE Trans Neural Netw 21(5):796–812
4. Liu YJ, Tong S, Li Y (2010) Adaptive neural network tracking control for a class of non-linear systems. Int J Syst Sci 41(2):143–158
5. Yang C, Ge SS, Lee TH (2009) Output feedback adaptive control of a class of nonlinear discrete-time systems with unknown control directions. Automatica 45(1):270–276
6. Li Z, Xia Y (2013) Adaptive neural network control of bilateral teleoperation with unsymmetrical stochastic delays and unmodeled dynamics. Int J Robust Nonlinear Control 24(11):1628–1652
7. Zhang Y, Yi C (2011) Zhang neural networks and neural-dynamic method. Nova Science Publishers, New York
8. Guo D, Zhang Y (2012) Zhang neural network, Getz-Marsden dynamic system, and discrete-time algorithms for time-varying matrix inversion with application to robots' kinematic control. Neurocomputing 97:22–32
9. Guo D, Qiu B, Ke Z, Yang Z, Zhang Y (2014) Case study of Zhang matrix inverse for different ZFs leading to different nets. In: Proceedings of the international joint conference on neural networks, pp 2764–2769

10. Wang J (1993) A recurrent neural network for real-time matrix inversion. Appl Math Comput 55(1):89–100
11. Zhang Y, Chen K, Tan H (2009) Performance analysis of gradient neural network exploited for online time-varying matrix inversion. IEEE Trans Autom Control 54(8):1940–1945
12. Zhang Y, Ge SS (2005) Design and analysis of a general recurrent neural network model for time-varying matrix inversion. IEEE Trans Neural Netw 16(6):1477–1490
13. Getz NH, Marsden JE (1995) A dynamic inverse for nonlinear maps. In: Proceedings of 34th IEEE conference on decision and control, pp 4218–4223
14. Getz NH, Marsden JE (1995) Joint-space tracking of workspace trajectories in continuous time. In: Proceedings of 34th IEEE conference on decision and control, pp 1001–1006
15. Getz NH, Marsden JE (1997) Dynamical methods for polar decomposition and inversion of matrices. Linear Algebra Appl 258:311–343
16. Song J, Yam Y (1998) Complex recurrent neural network for computing the inverse and pseudo-inverse of the complex matrix. Appl Math Comput 93(2–3):195–205
17. Zhang Y, Li Z, Li K (2011) Complex-valued Zhang neural network for online complex-valued time-varying matrix inversion. Appl Math Comput 217(24):10066–10073
18. Zhang Y, Guo D, Li F (2013) Different complex ZFs leading to different complex ZNN models for time-varying complex matrix inversion. In: Proceedings of 10th IEEE international conference on control and automation, pp 1330–1335

Chapter 13
Time-Varying Complex Matrix Generalized Inverse

Abstract In Chaps. 8 and 9, different ZD models based on different ZFs have been presented and investigated to solve for time-varying matrix (left and right) pseudoinverse in real domain. In this chapter, the ZD approach (i.e., different ZFs leading to different ZD models) is extended and exploited to solve for time-varying matrix generalized inverse (in most cases, the pseudoinverse) in complex domain. Specifically, by introducing five different complex ZFs, five different complex ZD models are proposed, generalized, developed, and investigated for time-varying complex matrix generalized inverse computation. Theoretical results of convergence analysis are presented to show the desirable properties of the complex ZD models. In addition, we discover the link between the proposed complex ZD models and the Getz-Marsden (G-M) dynamic system in complex domain. Computer simulation results further substantiate the efficacy of the proposed complex ZD models based on different complex ZFs on solving for time-varying complex matrix generalized inverse.

13.1 Introduction

As presented in Chaps. 8 and 9, the solution of generalized inverse (in most cases, the pseudoinverse, and also known as Moore-Penrose generalized inverse) is one of the basic problems encountered in a variety of science and engineering fields, e.g., robotics [1], signal processing [2], associative memories [3] and image restoration [4, 5]. Owing to its important roles, numerous efforts have been devoted to the fast solution of generalized inverse matrices. As a result, many algorithms/methods (including those ZD models presented in Chaps. 8 and 9) have been put forward by researchers [6–12] for constant and/or time-varying matrix generalize inverse computation. However, it is worth pointing out that these research works are confined to applying different numerical algorithms or neural dynamics to solving for matrix generalized inverse (or sometimes termed, matrix pseudoinverse) in real domain.

Besides, as presented in Chap. 12, in some situations complex-valued matrices may also occur, when the problem incorporates online frequency domain identification processes, or when the input signals contain both magnitude and phase information [13, 14]. The presence of a complex-valued matrix points to the need for

Y. Zhang and D. Guo, *Zhang Functions and Various Models*,
DOI 10.1007/978-3-662-47334-4_13

efficient online complex matrix inversion/pseudoinversion as well. In general, as for a complex-valued matrix $C \in \mathbb{C}^{m\times n}$, there are two cases, i.e., $m = n$ and $m \neq n$. Then, we have that $C^{-} \in \mathbb{C}^{n\times m}$ is known as the inverse of matrix C with $m = n$, and $C^{+} \in \mathbb{C}^{n\times m}$ is known as the generalized inverse of matrix C with $m \neq n$. Note that the investigations of solving for constant and time-varying complex square matrix inverse (i.e., corresponding to the situation of $m = n$) have been presented in [14, 15] and Chap. 12, respectively. Thus, in this chapter, we focus on solving for the generalized inverse (in most cases, the pseudoinverse) of time-varying complex matrix $C(t) \in \mathbb{C}^{m\times n}$ under the situation of $m \neq n$.

To lay a basis for further discussion, some necessary preliminaries of the time-varying complex matrix generalized inverse are given.

Definition 13.1 For a given time-varying complex matrix $C(t) \in \mathbb{C}^{m\times n}$ with $m \neq n$, if $Z(t) \in \mathbb{C}^{n\times m}$ satisfies at least one of the following four Penrose equations [16, 17]:

$$\begin{aligned} &C(t)Z(t)C(t) = C(t), \quad Z(t)C(t)Z(t) = Z(t),\\ &(C(t)Z(t))^{\mathrm{H}} = C(t)Z(t), \quad (Z(t)C(t))^{\mathrm{H}} = Z(t)C(t), \end{aligned}$$

where superscript $^{\mathrm{H}}$ denotes the conjugate transpose (also called Hermitian transpose) of a complex matrix, $Z(t)$ is called the time-varying complex generalized inverse of $C(t)$. If matrix $Z(t)$ satisfies all of the Penrose equations, then matrix $Z(t)$ is called the pseudoinverse of matrix $C(t)$, which is often denoted by $C^{+}(t)$. Note that the pseudoinverse $C^{+}(t)$ exists and is unique, while the generalized inverse is not unique usually.

In addition, if matrix $C(t)$ is of full-rank at any time instant t, i.e., $\operatorname{rank}(C(t)) = \min\{m, n\}$ with $t \in [0, \infty)$, we have the following theorem to obtain the time-varying pseudoinverse of matrix $C(t)$.

Theorem 13.1 *For a given time-varying matrix $C(t) \in \mathbb{C}^{m\times n}$ with $m \neq n$, if it satisfies that $\operatorname{rank}(C(t)) = \min\{m, n\}$ at any time instant t, then the unique time-varying pseudoinverse $C^{+}(t)$is given as follows [17–19]:*

$$C^{+}(t) = \begin{cases} C^{\mathrm{H}}(t)(C(t)C^{\mathrm{H}}(t))^{-1}, & \text{if } m < n,\\ (C^{\mathrm{H}}(t)C(t))^{-1}C^{\mathrm{H}}(t), & \text{if } m > n. \end{cases} \tag{13.1}$$

Besides, as for the unique time-varying pseudoinverse of a full-rank matrix $C(t)$, we have another important theorem as follows (which motivates us to define many more ZFs for time-varying complex matrix generalized inverse).

Theorem 13.2 *For a given time-varying matrix $C(t) \in \mathbb{C}^{m\times n}$ with $m \neq n$, if it satisfies that $\operatorname{rank}(C(t)) = \min\{m, n\}$ at any time instant t, then the unique time-varying pseudoinverse $C^{+}(t)$ is also given as follows:*

$$C^{+}(t)=\begin{cases}\lim\limits_{\mu\to 0}(C^{\mathrm{H}}(t)C(t)+\mu I)^{-1}C^{\mathrm{H}}(t), & \textit{if } m<n,\\ \lim\limits_{\mu\to 0}C^{\mathrm{H}}(t)(C(t)C^{\mathrm{H}}(t)+\mu I)^{-1}, & \textit{if } m>n,\end{cases}$$

where $\mu > 0 \in \mathbb{R}$.

Proof It can be generalized from the proof of Theorem 8.2. □

For simplicity, in this chapter, we only consider the smoothly time-varying full-rank complex matrix $C(t) \in \mathbb{C}^{m\times n}$ with $m < n$. This paper aims at finding $Z(t) \in \mathbb{C}^{n\times m}$ such that at least one of the Penrose equations holds true at any time instant $t \in [0, +\infty)$, i.e., obtaining the complex generalized inverse (in most cases, the pseudoinverse) of matrix $C(t)$. Note that, in the case of $m > n$, the complex generalized inverse of matrix $C(t)$ could be obtained in a similar way, and is thus omitted due to similarity and space limitation.

More specifically, focusing on solving for the generalized inverse of time-varying complex matrix $C(t)$ with $m < n$, we propose, generalize, develop, and investigate five different complex ZD models by defining five different complex ZFs. It is then theoretically proved that the proposed complex ZD models (globally) exponentially converge to the theoretical time-varying generalized inverse. Moreover, we discover the link between the proposed complex ZD models and the Getz-Marsden (G-M) dynamic system [20] in the complex domain. Through illustrative computer-simulation examples, the efficacy of the proposed complex ZD models for time-varying complex matrix generalized inverse computation is well substantiated.

13.2 Complex ZFs and ZD Models

In this section, five different complex ZD models based on five different complex ZFs are constructed to solve for the time-varying complex generalized inverse (in most cases, the pseudoinverse). In addition, their excellent convergence performances are analyzed in detail.

According to the ZD design formula (12.2), different complex ZFs can lead to different complex ZD models for solving the same time-varying complex-valued problem. Especially, to solve for the time-varying complex generalized inverse, we define the following five different complex ZFs as the fundamental error-monitoring functions:

$$E(t)=Z(t)C(t)C^{\mathrm{H}}(t)-C^{\mathrm{H}}(t)\in\mathbb{C}^{n\times m}, \tag{13.2}$$

$$E(t)=C^{\mathrm{H}}(t)C(t)Z(t)-C^{\mathrm{H}}(t)\in\mathbb{C}^{n\times m}, \tag{13.3}$$

$$E(t)=C(t)Z(t)-I\in\mathbb{C}^{m\times m}, \tag{13.4}$$

$$E(t)=Z(t)C(t)-I\in\mathbb{C}^{n\times n}, \tag{13.5}$$

$$E(t)=C(t)-Z^{+}(t)\in\mathbb{C}^{m\times n}. \tag{13.6}$$

13.2.1 The First Complex ZD Model

Considering complex ZF (13.2), we have the following derivation:

$$\dot{E}(t) = \dot{Z}(t)C(t)C^{\mathrm{H}}(t) + Z(t)\left(\dot{C}(t)C^{\mathrm{H}}(t) + C(t)\dot{C}^{\mathrm{H}}(t)\right) - \dot{C}^{\mathrm{H}}(t).$$

Then, adopting the ZD design formula (12.2), we can derive the corresponding dynamic equation of the first complex ZD model as

$$\begin{aligned}\dot{Z}(t)C(t)C^{\mathrm{H}}(t) = \dot{C}^{\mathrm{H}}(t) - Z(t)\left(\dot{C}(t)C^{\mathrm{H}}(t) + C(t)\dot{C}^{\mathrm{H}}(t)\right)\\ -\gamma\left(Z(t)C(t)C^{\mathrm{H}}(t) - C^{\mathrm{H}}(t)\right).\end{aligned} \tag{13.7}$$

In other words, we obtain complex ZD model (13.7) based on complex ZF (13.2) to solve for the time-varying complex generalized inverse (specifically, the pseudoinverse). By following complex ZD model (13.7), the *ij*th neuron's dynamic equation can be presented in the following form:

$$\dot{z}_{ij} = \sum_{l=1}^{m} \dot{z}_{il}a_{lj} - \gamma\left(\sum_{l=1}^{m} z_{il}b_{lj} - c_{ji}^{*}\right) - \sum_{l=1}^{m} z_{il}d_{lj} + \dot{c}_{ji}^{*},$$

where c_{ji}, a_{lj}, b_{lj} and d_{lj} denote the corresponding elements of matrices C, $A = I - CC^{\mathrm{H}}$, $B = CC^{\mathrm{H}}$ and $D = \dot{C}C^{\mathrm{H}} + C\dot{C}^{\mathrm{H}}$, respectively, and the operator $*$ denotes complex conjugate. Then, the neuron-connection architecture of complex ZD model (13.7) is depicted in Fig. 13.1, and the specific structure of the ith row of neurons is illustrated in Fig. 13.2. Figures 13.1 and 13.2 well show that complex ZD model (13.7) is a kind of Hopfield-type recurrent neural networks which can be implemented finally on analog circuits such as very large-scale integration [11, 21, 22].

To lay a basis for discussion, an important theorem is presented below.

Theorem 13.3 *For any time-varying complex matrix $C(t) \in \mathbb{C}^{m\times n}$, we have [23]*

$$\frac{\mathrm{d}C^{\mathrm{H}}(t)}{\mathrm{d}t} = \left(\frac{\mathrm{d}C(t)}{\mathrm{d}t}\right)^{\mathrm{H}},$$

which, via a simpler notation of $\mathrm{d}C^{\mathrm{H}}(t)/\mathrm{d}t$, can be rewritten as $\dot{C}^{\mathrm{H}}(t) = (\dot{C}(t))^{\mathrm{H}}$. Especially, for a scalar complex variable $c(t) \in \mathbb{C}$, we have $\dot{c}^{*}(t) = (\dot{c}(t))^{*}$.

For complex ZD model (13.7), we have the following theoretical result on its global exponential convergence performance.

Theorem 13.4 *Given a smoothly time-varying complex matrix $C(t) \in \mathbb{C}^{m\times n}$ (with $m < n$) of full rank, the state matrix $Z(t) \in \mathbb{C}^{n\times m}$ of complex ZD model (13.7), starting from an initial state $Z(0)$, globally and exponentially converges to the theoretical time-varying generalized inverse [specifically, the pseudoinverse $C^{+}(t) \in \mathbb{C}^{n\times m}$] of matrix $C(t)$.*

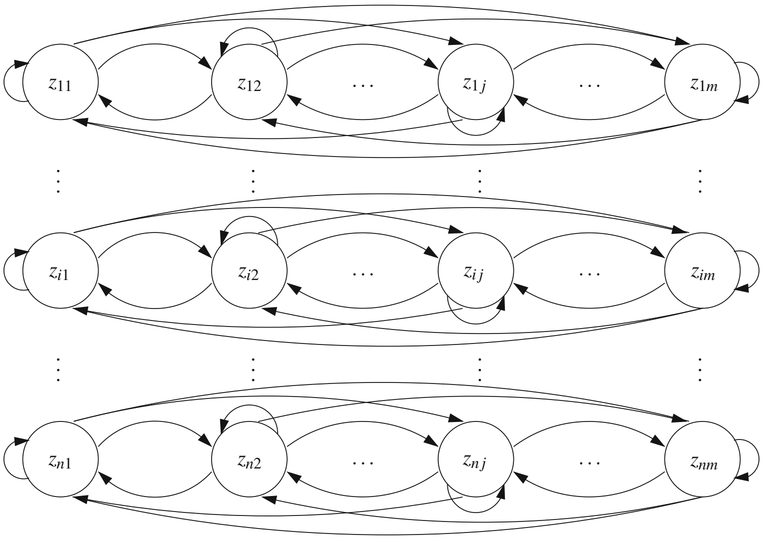


Fig. 13.1 Neuron-connection architecture of complex ZD model (13.7) for time-varying complex generalized inverse computation

Proof Let $\tilde{Z}(t) = Z(t) - C^{+}(t)$ denote the difference between the solution $Z(t)$ generated by complex ZD model (13.7) and the theoretical pseudoinverse $C^{+}(t)$. Following from $C^{+}(t)C(t)C^{\mathrm{H}}(t) - C^{\mathrm{H}}(t) = 0$, its time derivative is depicted as

$$\dot{C}^{+}(t)C(t)C^{\mathrm{H}}(t) + C^{+}(t)\left(\dot{C}(t)C^{\mathrm{H}}(t) + C(t)\dot{C}^{\mathrm{H}}(t)\right) - \dot{C}^{\mathrm{H}}(t) = 0.$$

Substituting $C^{+}(t) = Z(t) - \tilde{Z}(t)$ into the above identity, we have

$$\begin{aligned}&\dot{\tilde{Z}}(t)C(t)C^{\mathrm{H}}(t) + \tilde{Z}(t)\left(\dot{C}(t)C^{\mathrm{H}}(t) + C(t)\dot{C}^{\mathrm{H}}(t)\right) = \\ &\dot{Z}(t)C(t)C^{\mathrm{H}}(t) + Z(t)\left(\dot{C}(t)C^{\mathrm{H}}(t) + C(t)\dot{C}^{\mathrm{H}}(t)\right) - \dot{C}^{\mathrm{H}}(t).\end{aligned}$$

Using complex ZD model equation (13.7), with $Z(t) = \tilde{Z}(t) + C^{+}(t)$, it follows that $\tilde{Z}(t)$ is the solution to the ensuing dynamics with the initial state $\tilde{Z}(0) = Z(0) - C^{+}(0)$,

$$\dot{\tilde{Z}}(t)C(t)C^{\mathrm{H}}(t) + \tilde{Z}(t)\left(\dot{C}(t)C^{\mathrm{H}}(t) + C(t)\dot{C}^{\mathrm{H}}(t)\right) = -\gamma\tilde{Z}(t)C(t)C^{\mathrm{H}}(t). \quad (13.8)$$

Since $E(t) = \tilde{Z}(t)C(t)C^{\mathrm{H}}(t)$, (13.8) can thus be rewritten as $\dot{E}(t) = -\gamma E(t)$, which is a compact matrix form of the following set of $n \times m$ equations:

$$\dot{e}_{ij}(t) = -\gamma e_{ij}(t), \ \forall i \in \{1, 2, \ldots, n\} \text{ and } j \in \{1, 2, \ldots, m\}. \quad (13.9)$$

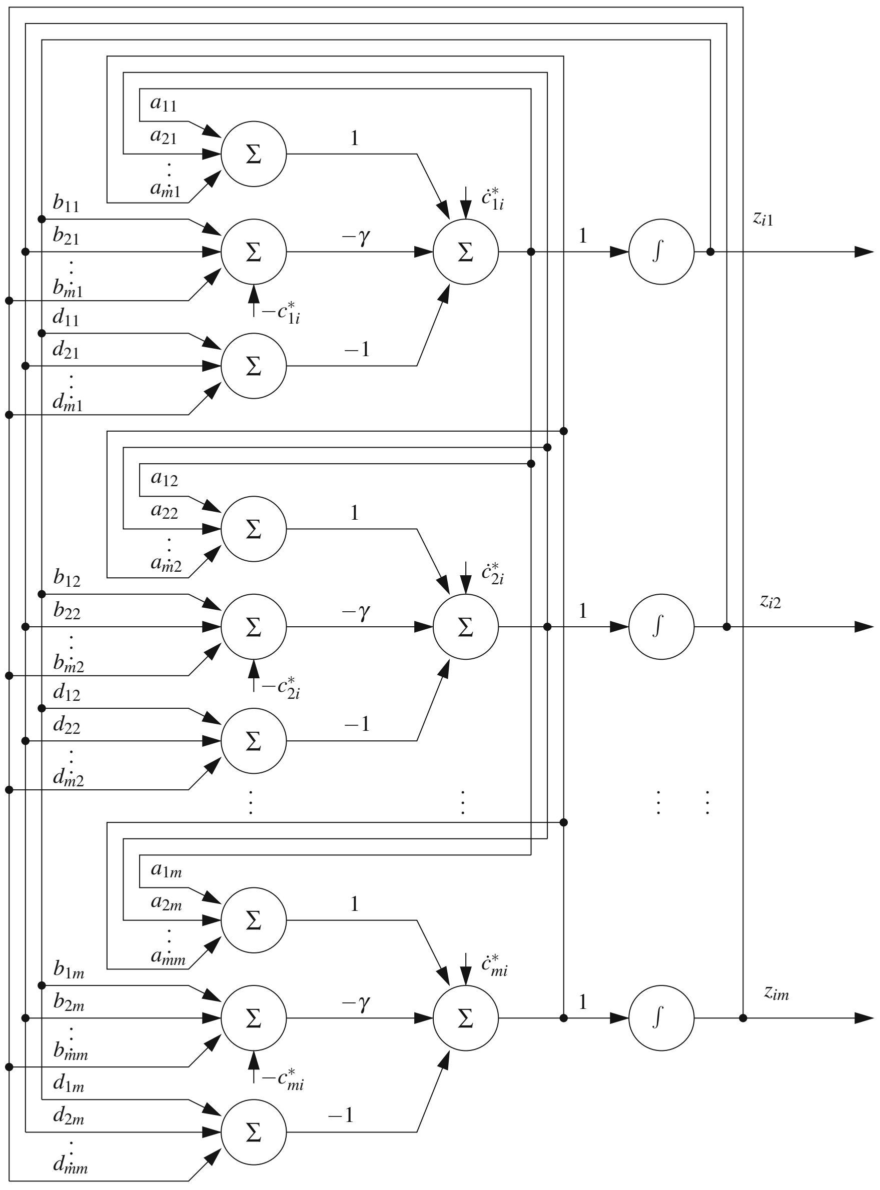


Fig. 13.2 Structure of the ith row of neurons in complex ZD model (13.7) for time-varying complex generalized inverse computation

Evidently, we can define a Lyapunov function candidate $v_{ij} = e_{ij}e_{ij}^*/2 \geqslant 0$ for the ijth subsystem (13.9), which is positive-definite, i.e., $v_{ij} > 0$ for $e_{ij} \neq 0$ and $v_{ij} = 0$ for $e_{ij} = 0$. Then, we have its time derivative

$$\frac{\mathrm{d}v_{ij}(t)}{\mathrm{d}t} = \frac{1}{2}\left(\dot{e}_{ij}e_{ij}^* + e_{ij}\dot{e}_{ij}^*\right).$$

Adopting Theorem 13.3 and (13.9), we obtain

$$\frac{\mathrm{d}v_{ij}(t)}{\mathrm{d}t} = \frac{1}{2}\left((-\gamma e_{ij})e_{ij}^* + e_{ij}(-\gamma e_{ij})^*\right) = -\gamma e_{ij}e_{ij}^*.$$

Apparently, $\dot{v}_{ij}$ is negative-definite, i.e., $\dot{v}_{ij} < 0$ for $e_{ij} \neq 0$ and $\dot{v}_{ij} = 0$ for $e_{ij} = 0$. In addition, if $|e_{ij}| \to \infty$, the Lyapunov function candidate $v_{ij} = |e_{ij}|^2/2 \to \infty$. By the Lyapunov stability theory, $e_{ij}(t)$ globally converges to zero for any $i \in \{1, 2, \ldots, n\}$ and $j \in \{1, 2, \ldots, m\}$. Thus, in view of $E(t) = Z(t)C(t)C^{\mathrm{H}}(t) - C^{\mathrm{H}}(t)$ and the nonsingularity of $C(t)C^{\mathrm{H}}(t)$, we have $Z(t) \to C^{\mathrm{H}}(t)(C(t)C^{\mathrm{H}}(t))^{-1} \in \mathbb{C}^{n\times m}$ as $t \to \infty$. Then, in view of $m < n$ and based on Theorem 13.1, the state matrix $Z(t)$ of (13.7) globally converges to the theoretical time-varying generalized inverse [specifically, the pseudoinverse $C^+(t) \in \mathbb{C}^{n\times m}$] starting from a randomly-generated initial state $Z(0)$. Next, we are going to prove the exponential convergence performance of complex ZD model (13.7).

In view of (13.9), we can obtain its analytic solution in the compact matrix form:

$$E(t) = E(0)\exp(-\gamma t).$$

Thus, we further have

$$\|E(t)\|_{\mathrm{F}} = \|E(0)\|_{\mathrm{F}}\exp(-\gamma t).$$

Evidently, as $t \to \infty$, $\|E(t)\|_{\mathrm{F}}$ exponentially converges to 0 with rate γ, which means that, starting from any randomly-generated initial state $Z(0)$, state matrix $Z(t)$ of complex ZD model (13.7) exponentially converges to the theoretical time-varying generalized inverse [specifically, the pseudoinverse $C^+(t)$] with rate $\gamma > 0$. The proof on global and exponential convergence of complex ZD model (13.7) is thus complete. □

For further investigation and illustration, we can also make use of other complex ZFs [i.e., complex ZFs (13.3) through (13.6)] to construct other types of complex ZD models. Thus, it can provide many more models for researchers to choose.

13.2.2 The Second Complex ZD Model

For complex ZF (13.3), as $m < n$, $C^{\mathrm{H}}(t)C(t) \in \mathbb{C}^{n\times n}$ is singular. Thus, we can add a bias term $\lambda I \in \mathbb{R}^{n\times n}$ to $C^{\mathrm{H}}(t)C(t)$, with $\lambda > 0 \in \mathbb{R}$. This method is known as

Tikhonov regularization method [24]. Then, complex ZF (13.3) is modified as

$$E(t) = \left(C^{\mathrm{H}}(t)C(t) + \lambda I\right) Z(t) - C^{\mathrm{H}}(t). \tag{13.10}$$

For modified ZF (13.10), we have its time derivative

$$\dot{E}(t) = \left(C^{\mathrm{H}}(t)C(t) + \lambda I\right) \dot{Z}(t) + \left(\dot{C}^{\mathrm{H}}(t)C(t) + C^{\mathrm{H}}(t)\dot{C}(t)\right) Z(t) - \dot{C}^{\mathrm{H}}(t).$$

Following the ZD design formula (12.2), we obtain the dynamic equation of the second complex ZD model as

$$\begin{aligned}\left(C^{\mathrm{H}}(t)C(t) + \lambda I\right) \dot{Z}(t) = \dot{C}^{\mathrm{H}}(t) - \left(\dot{C}^{\mathrm{H}}(t)C(t) + C^{\mathrm{H}}(t)\dot{C}(t)\right) Z(t)\\ -\gamma \left(\left(C^{\mathrm{H}}(t)C(t) + \lambda I\right) Z(t) - C^{\mathrm{H}}(t)\right).\end{aligned} \tag{13.11}$$

Note that the parameter λ should be set appropriately small, in other words, λ should be sufficiently close to 0. Similarly, after presenting complex ZD model (13.11) for solving for the time-varying complex generalized inverse (specifically, the pseudoinverse), we come to prove its convergence performance through the following important theorem.

Theorem 13.5 *Given a smoothly time-varying complex matrix $C(t) \in \mathbb{C}^{m\times n}$ (with $m < n$) of full rank, the state matrix $Z(t) \in \mathbb{C}^{n\times m}$ of complex ZD model (13.11), starting from an initial state $Z(0)$, globally and exponentially converges to the theoretical time-varying generalized inverse [specifically, the pseudoinverse $C^{+}(t) \in \mathbb{C}^{n\times m}$] of matrix $C(t)$.*

Proof Since complex ZD model (13.11) is derived using the standard ZD design method similar to the aforementioned first complex ZD model, its modified ZF (13.10) satisfies relation (12.2), which means that $E(t) = (C^{\mathrm{H}}(t)C(t) + \lambda I)Z(t) - C^{\mathrm{H}}(t)$ can globally and exponentially converge to zero from an initial value. That is to say, as $t \to \infty$, we have $Z(t) \to (C^{\mathrm{H}}(t)C(t) + \lambda I)^{-1}C^{\mathrm{H}}(t) \in \mathbb{C}^{n\times m}$. In view of $\lambda \to 0$ and $m < n$, then based on Theorem 13.2, the state matrix $Z(t)$ globally and exponentially converges to the theoretical time-varying generalized inverse, specifically, the pseudoinverse $C^{+}(t)$. The proof on global and exponential convergence performance of complex ZD model (13.11) is thus complete. □

13.2.3 The Third Complex ZD Model

Combining the ZD design formula (12.2) and complex ZF (13.4), we can have

$$\dot{C}(t)Z(t) + C(t)\dot{Z}(t) = -\gamma \left(C(t)Z(t) - I\right),$$

and then

$$C(t)\dot{Z}(t) = -\dot{C}(t)Z(t) - \gamma\left(C(t)Z(t) - I\right).$$

For the purpose of computation and simulation, by left multiplying $C^{\mathrm{H}}(t)$ both sides of the above equation, we obtain

$$C^{\mathrm{H}}(t)C(t)\dot{Z}(t) = -C^{\mathrm{H}}(t)\dot{C}(t)Z(t) - \gamma\left(C^{\mathrm{H}}(t)C(t)Z(t) - C^{\mathrm{H}}(t)\right). \quad (13.12)$$

Note that, in (13.12), $C^{\mathrm{H}}(t)C(t)$ is singular (in view of $m < n$). Hence, to make (13.12) more computable, we can similarly adopt the Tikhonov regularization method [24], i.e., add a bias term λI with $\lambda \to 0$ to $C^{\mathrm{H}}(t)C(t)$. As a result, the dynamic equation of the third complex ZD model is presented as

$$\begin{aligned}\left(C^{\mathrm{H}}(t)C(t) + \lambda I\right)\dot{Z}(t) = {} & -C^{\mathrm{H}}(t)\dot{C}(t)Z(t) \\ & - \gamma\left(\left(C^{\mathrm{H}}(t)C(t) + \lambda I\right)Z(t) - C^{\mathrm{H}}(t)\right). \quad (13.13)\end{aligned}$$

That is to say, based on complex ZF (13.4), we obtain complex ZD model (13.13) to solve for the time-varying complex generalized inverse (specifically, the pseudoinverse). Similarly, the important theorem about the convergence performance of complex ZD model (13.13) is given as follows.

Theorem 13.6 *Given a smoothly time-varying complex matrix $C(t) \in \mathbb{C}^{m\times n}$ (with $m < n$) of full rank, the state matrix $Z(t) \in \mathbb{C}^{n\times m}$ of complex ZD model (13.13), starting from an initial state $Z(0)$, globally and exponentially converges to the theoretical time-varying generalized inverse [specifically, the pseudoinverse $C^{+}(t) \in \mathbb{C}^{n\times m}$] of matrix $C(t)$.*

Proof The convergence of complex ZD model (13.13) can be proven in a way similar to the proofs of Theorems 13.4 and 13.5, and thus it is omitted here. □

13.2.4 The Fourth Complex ZD Model

With the ZD design formula (12.2) and complex ZF (13.5), we have

$$\dot{Z}(t)C(t) + Z(t)\dot{C}(t) = -\gamma\left(Z(t)C(t) - I\right),$$

and then

$$\dot{Z}(t)C(t) = -Z(t)\dot{C}(t) - \gamma\left(Z(t)C(t) - I\right). \quad (13.14)$$

Similarly, to make (13.14) more computable, we right multiply $C^{\mathrm{H}}(t)$ both sides of (13.14), and obtain the dynamic equation of the fourth complex ZD model as

$$\dot{Z}(t)C(t)C^{\mathrm{H}}(t) = -Z(t)\dot{C}(t)C^{\mathrm{H}}(t) - \gamma\left(Z(t)C(t)C^{\mathrm{H}}(t) - C^{\mathrm{H}}(t)\right). \tag{13.15}$$

Thus, based on complex ZF (13.5), we have complex ZD model (13.15) to solve for the time-varying complex generalized inverse (specifically, pseudoinverse). Similar to the previous ZD models, we have the following important result about the convergence performance of complex ZD model (13.15). That is, given a smoothly time-varying complex matrix $C(t) \in \mathbb{C}^{m\times n}$ (with $m < n$) of full rank, the state matrix $Z(t) \in \mathbb{C}^{n\times m}$ of complex ZD model (13.15), starting from an initial state $Z(0)$, globally and exponentially converges to the theoretical time-varying generalized inverse, specifically, the pseudoinverse $C^{+}(t) \in \mathbb{C}^{n\times m}$.

13.2.5 The Fifth Complex ZD Model

Before constructing the fifth complex ZD model, an important corollary (being an extension of Corollary 5.1 from the real domain to the complex domain) is presented here to lay a basis for discussion.

Corollary 13.1 *For a given time-varying complex matrix $C(t) \in \mathbb{C}^{m\times n}$ (with $m < n$) and its time-varying pseudoinverse $C^{+}(t)$, we approximately have $\dot{C}^{+}(t) = -C^{+}(t)\dot{C}(t)C^{+}(t)$.*

Proof It can also be generalized from the proof of Theorem 4.1 in Chap. 4. □

Then, based on the ZD design formula (12.2) and complex ZF (13.6), we can have

$$\dot{C}(t) - \dot{Z}^{+}(t) = -\gamma\left(C(t) - Z^{+}(t)\right).$$

By adopting Corollary 13.1, the above equation can be further rewritten as

$$\begin{aligned}\dot{C}(t) + Z^{+}(t)\dot{Z}(t)Z^{+}(t) &= -\gamma\left(C(t) - Z^{+}(t)\right),\\ Z^{+}(t)\dot{Z}(t)Z^{+}(t) &= -\dot{C}(t) - \gamma\left(C(t) - Z^{+}(t)\right).\end{aligned} \tag{13.16}$$

Reformulating (13.16), we have the following dynamic equation of the new complex ZD model aiming at solving for the time-varying complex generalized inverse:

$$\dot{Z}(t) = -Z(t)\dot{C}(t)Z(t) - \gamma\left(Z(t)C(t)Z(t) - Z(t)\right), \tag{13.17}$$

Thus, we obtain complex ZD model (13.17) based on complex ZF (13.6). Note that complex ZD model (13.17) is also the Getz and Marsden (G-M) dynamic system [20] for the time-varying complex generalized inverse computation. In other words, the

G-M dynamic system could be generalized to the case of complex matrices and is a special case of the complex ZD models. This is quite a novel result beyond our previous work. Following the literature [20], we have the theoretical results of the convergence performance of complex ZD model (13.17). That is, given a smoothly time-varying complex matrix $C(t) \in \mathbb{C}^{m\times n}$ of full rank, if initial state $Z(0)$ satisfies $\|Z(0) - C^{+}(0)\|_{\mathrm{F}} \leqslant \beta < \infty$ and $\beta \in \mathbb{R}$ is sufficiently small, then $C(t)Z(t) - I \to 0$ as $t \to \infty$, i.e., the state matrix $Z(t) \in \mathbb{C}^{n\times m}$ of complex ZD model (13.17) exponentially converges to the theoretical time-varying generalized inverse of matrix $C(t)$. It is worth noting that the initial condition of complex ZD model (13.17) should be chosen as $Z(0) \approx C^{+}(0)$ [i.e., $Z(0)$ should be sufficiently close to $C^{+}(0)$].

In summary, we have constructed five different complex ZD models (13.7), (13.11), (13.13), (13.15), and (13.17) by defining five different complex ZFs [i.e., complex ZFs (13.2)–(13.6)] to solve for time-varying complex generalized inverse (in most cases, the pseudoinverse). For readers' convenience and also for comparison purpose, we summarize these complex ZFs and the corresponding complex ZD models in Table 13.1.

Remark 13.1 Five complex ZFs have been elaborately constructed to obtain five different complex ZD models. There exist clear differences among such complex ZD models. Specifically, the dynamic equations, model complexities and convergence performance differ from each other. For instance, the fifth complex ZD model (13.17) has the simplest network structure which can be more readily implemented, whereas the first complex ZD model (13.7) has better global convergence performance. Therefore, in practical applications, the practitioner could find and choose the most suitable complex ZF and the corresponding complex ZD model in accordance with specific request.

Table 13.1 Different complex ZFs resulting in different complex ZD models for time-varying complex generalized inverse (in most cases, the pseudoinverse) computation

Complex ZF	Complex ZD model
(13.2)	$\dot{Z}(t)C(t)C^{\mathrm{H}}(t) = -\gamma(Z(t)C(t)C^{\mathrm{H}}(t) - C^{\mathrm{H}}(t)) - Z(t)(\dot{C}(t)C^{\mathrm{H}}(t) + C(t)\dot{C}^{\mathrm{H}}(t)) + \dot{C}^{\mathrm{H}}(t)$
(13.3)	$(C^{\mathrm{H}}(t)C(t) + \lambda I)\dot{Z}(t) = \dot{C}^{\mathrm{H}}(t) - (\dot{C}^{\mathrm{H}}(t)C(t) + C^{\mathrm{H}}(t)\dot{C}(t))Z(t) - \gamma((C^{\mathrm{H}}(t)C(t) + \lambda I)Z(t) - C^{\mathrm{H}}(t))$
(13.4)	$(C^{\mathrm{H}}(t)C(t) + \lambda I)\dot{Z}(t) = -C^{\mathrm{H}}(t)\dot{C}(t)Z(t) - \gamma((C^{\mathrm{H}}(t)C(t) + \lambda I)Z(t) - C^{\mathrm{H}}(t))$
(13.5)	$\dot{Z}(t)C(t)C^{\mathrm{H}}(t) = -Z(t)\dot{C}(t)C^{\mathrm{H}}(t) - \gamma(Z(t)C(t)C^{\mathrm{H}}(t) - C^{\mathrm{H}}(t))$
(13.6)	$\dot{Z}(t) = -Z(t)\dot{C}(t)Z(t) - \gamma(Z(t)C(t)Z(t) - Z(t))$

13.3 Illustrative Examples

In this section, the related simulation techniques are presented, and four illustrative examples are given to substantiate the efficacy of the proposed complex ZD models [i.e., (13.7), (13.11), (13.13), (13.15), and (13.17)] on solving for time-varying complex generalized inverse (in most cases, the pseudoinverse).

Kronecker product and vectorization In the previous sections, we have developed five complex ZD models (13.7), (13.11), (13.13), (13.15), and (13.17) for time-varying complex generalized inverse computation. Note that all the proposed complex ZD models are described in matrix form, which cannot be directly simulated. Thus, the Kronecker product and vectorization techniques [11, 25] are needed to transform such matrix-form differential equations to vector-form differential equations for simulative purposes. Note that, for presentation convenience, $B(t) = C^{\mathrm{H}}(t)C(t) + \lambda I$ is introduced [for complex ZD models (13.11) and (13.13)].

- For complex ZD model (13.7), based on the Kronecker product (denoted by the symbol of "⊗") and vectorization techniques, we can transform such a complex ZD model into the following vector-form differential equation:

$$\begin{aligned}\left((CC^{\mathrm{H}})^{\mathrm{T}} \otimes I\right) \mathrm{vec}(\dot{Z}) = \mathrm{vec}(\dot{C}^{\mathrm{H}}) - \left((\dot{C}C^{\mathrm{H}})^{\mathrm{T}} \otimes I + (C\dot{C}^{\mathrm{H}})^{\mathrm{T}} \otimes I\right) \mathrm{vec}(Z) \\ -\gamma \left(\left((CC^{\mathrm{H}})^{\mathrm{T}} \otimes I\right) \mathrm{vec}(Z) - \mathrm{vec}(C^{\mathrm{H}})\right).\end{aligned}$$

- In view of $B(t) = C^{\mathrm{H}}(t)C(t) + \lambda I$, complex ZD model (13.11) is rewritten as

$$B(t)\dot{Z}(t) = \dot{C}^{\mathrm{H}}(t) - \left(\dot{C}^{\mathrm{H}}(t)C(t) + C^{\mathrm{H}}(t)\dot{C}(t)\right) Z(t) - \gamma \left(B(t)Z(t) - C^{\mathrm{H}}(t)\right).$$

  Therefore, similar to (13.7), we obtain the vector form of (13.11) as follows:

$$\begin{aligned}(I \otimes B)\, \mathrm{vec}(\dot{Z}) = \mathrm{vec}(\dot{C}^{\mathrm{H}}) - \left((I \otimes \dot{C}^{\mathrm{H}}C) + (I \otimes C^{\mathrm{H}}\dot{C})\right) \mathrm{vec}(Z) \\ -\gamma \left((I \otimes B)\, \mathrm{vec}(Z) - \mathrm{vec}(C^{\mathrm{H}})\right).\end{aligned}$$

- Similarly, for complex ZD model (13.13), we can have its vector form as

$$(I \otimes B)\, \mathrm{vec}(\dot{Z}) = -(I \otimes C^{\mathrm{H}}\dot{C})\, \mathrm{vec}(Z) - \gamma \left((I \otimes B)\, \mathrm{vec}(Z) - \mathrm{vec}(C^{\mathrm{H}})\right).$$

- Considering complex ZD model (13.15), we similarly have its vector form as

$$\begin{aligned}\left((CC^{\mathrm{H}})^{\mathrm{T}} \otimes I\right) \mathrm{vec}(\dot{Z}) = -\left((\dot{C}C^{\mathrm{H}})^{\mathrm{T}} \otimes I\right) \mathrm{vec}(Z) \\ -\gamma \left(\left((CC^{\mathrm{H}})^{\mathrm{T}} \otimes I\right) \mathrm{vec}(Z) - \mathrm{vec}(C^{\mathrm{H}})\right).\end{aligned}$$

- For complex ZD model (13.17), we can also obtain its vector form as

$$\mathrm{vec}(\dot{Z}) = -(I \otimes Z\dot{C})\, \mathrm{vec}(Z) - \gamma \left((I \otimes ZC)\, \mathrm{vec}(Z) - \mathrm{vec}(Z)\right).$$

Besides, it is worth pointing out here that, in MATLAB, the Kronecker product can be realized by using the routine "kron" [i.e., "$Z \otimes C$" is realized by the code "`kron(Z,C)`"], and "vec(Z)" is realized by the code "`reshape(Z,n*m,1)`".

Therefore, based on the aforementioned vectorization technique, the following four computer simulation examples are illustrated to substantiate the efficacy of the proposed complex ZD models (13.7), (13.11), (13.13), (13.15), and (13.17) on solving for time-varying complex generalized inverse.

Example 13.1 In this example, we consider the time-varying full-rank complex matrix $C(t)$ as follows:

$$C(t) = \begin{bmatrix} i\sin(3t) & i\cos(3t) & -i\sin(3t) \\ -i\cos(3t) & i\sin(3t) & i\cos(3t) \end{bmatrix} \in \mathbb{C}^{2\times 3}. \tag{13.18}$$

For checking the correctness of the ZD solution, according to (13.1), we can obtain the theoretical time-varying pseudoinverse of matrix $C(t)$ in (13.18) as

$$C^{+}(t) = \begin{bmatrix} -0.5i\sin(3t) & 0.5i\cos(3t) \\ -i\cos(3t) & -i\sin(3t) \\ 0.5i\sin(3t) & -0.5i\cos(3t) \end{bmatrix} \in \mathbb{C}^{3\times 2}.$$

Since we have obtained theoretical pseudoinverse $C^{+}(t)$, we can use it as an analytic theoretical solution to verify the correctness of the solution synthesized by complex ZD model (13.7). As illustrated in Fig. 13.3, starting from a randomly-generated initial state $Z(0) \in \mathbb{C}^{3\times 2}$, the state matrix $Z(t) \in \mathbb{C}^{3\times 2}$ of complex ZD model (13.7) with $\gamma = 100$ can converge to the theoretical pseudoinverse $C^{+}(t)$ rapidly and accurately within a rather short time. In addition, we show the residual errors $\|E(t)\| = \|Z(t)C(t)C^{\mathrm{H}}(t) - C^{\mathrm{H}}(t)\|_{\mathrm{F}}$ synthesized by the proposed complex ZD model (13.7) starting from 10 randomly-generated initial states. From Fig. 13.4, we can further find that the residual errors of (13.7) all diminish to zero within around 0.06 s. These simulation results demonstrate the efficacy of complex ZD model (13.7) on solving for time-varying complex generalized inverse (specifically, the pseudoinverse).

Example 13.2 In this example, we verify the efficacy of the proposed complex ZD models (13.11) and (13.13) use a more general complex matrix. Let us consider the following time-varying complex matrix:

$$C(t) = \begin{bmatrix} \exp(4it) & i\exp(4it) & \exp(-4it) \\ i\exp(4it) & \exp(4it) & i\exp(4it) \end{bmatrix} \in \mathbb{C}^{2\times 3}. \tag{13.19}$$

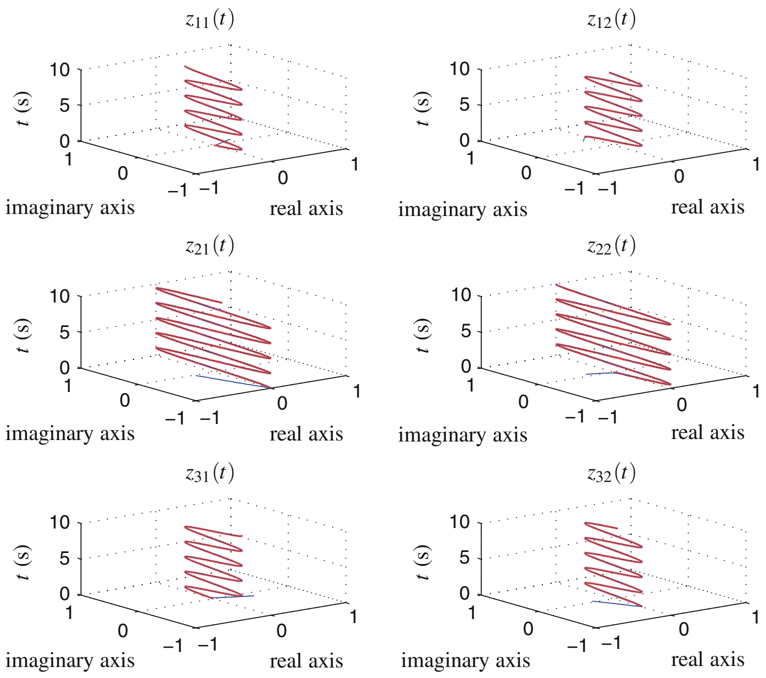


Fig. 13.3 State trajectories of complex ZD model (13.7) with $\gamma = 100$, where *dash-dotted curves* denote the theoretical time-varying pseudoinverse $C^{+}(t)$ in Example 13.1 and *solid curves* denote the solution computed by complex ZD model (13.7)

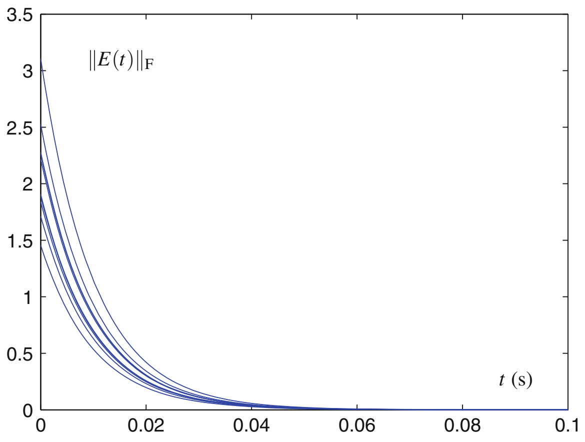


Fig. 13.4 Residual errors $\|E(t)\|_{\mathrm{F}} = \|Z(t)C(t)C^{\mathrm{H}}(t) - C^{\mathrm{H}}(t)\|_{\mathrm{F}}$ synthesized by complex ZD model (13.7) with $\gamma = 100$ for the time-varying pseudoinverse of matrix $C(t)$ in (13.18)

It follows from (13.1) that the theoretical time-varying pseudoinverse of matrix $C(t)$ in (13.19) is

$$C^{+}(t) = \begin{bmatrix} \frac{3}{8}\exp(-4it) - \frac{1}{8}\exp(4it) & -\frac{3}{8}i\exp(-4it) + \frac{1}{8}i\exp(-12it) \\ -\frac{3}{8}i\exp(-4it) - \frac{1}{8}i\exp(4it) & \frac{3}{8}\exp(-4it) + \frac{1}{8}\exp(-12it) \\ \frac{1}{4}\exp(4it) & -\frac{1}{4}i\exp(-4it) \end{bmatrix} \in \mathbb{C}^{3\times 2}.$$

Note that, in this example, the complex matrix $C(t)$ in (13.19) is more general since its elements have both real and imaginary parts. Furthermore, the variation frequency of such a complex matrix is greater than that of Example 13.1. Figures 13.5 and 13.6, respectively, illustrate the neural state $Z(t)$ of complex ZD models (13.11) and (13.13) by using $\gamma = 100$ and $\lambda = 10^{-3}$, with the residual errors $\|E(t)\|_{\mathrm{F}} = \|Z(t)C(t)C^{\mathrm{H}}(t) - C^{\mathrm{H}}(t)\|_{\mathrm{F}}$ shown in Fig. 13.7. As seen from Figs. 13.5 and 13.6, starting from a randomly-generated initial state $Z(0)$, the neural states $Z(t)$ of complex ZD models (13.11) and (13.13) both converge to the theoretical time-varying pseudoinverse $C^{+}(t)$. In addition, from Fig. 13.7, we can see that residual errors $\|E(t)\|_{\mathrm{F}}$ of (13.11) and (13.13) all converge to zero. Therefore, the efficacy of complex ZD models (13.11) and (13.13) on solving for the time-varying complex generalized inverse (specifically, the pseudoinverse) is also substantiated.

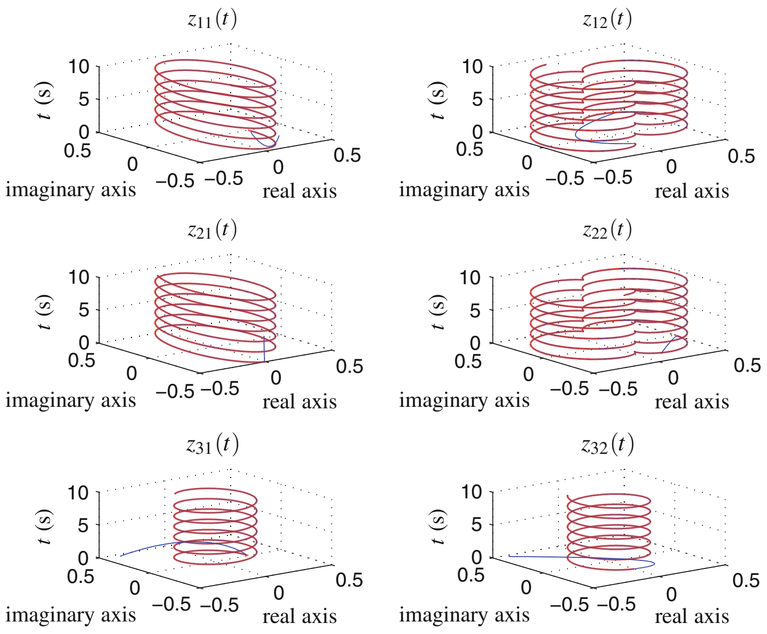


Fig. 13.5 State trajectories of complex ZD model (13.11) with $\gamma = 100$ and $\lambda = 10^{-3}$, where *dash-dotted curves* denote the theoretical time-varying pseudoinverse $C^{+}(t)$ in Example 13.2 and *solid curves* denote the solution computed by complex ZD model (13.11)

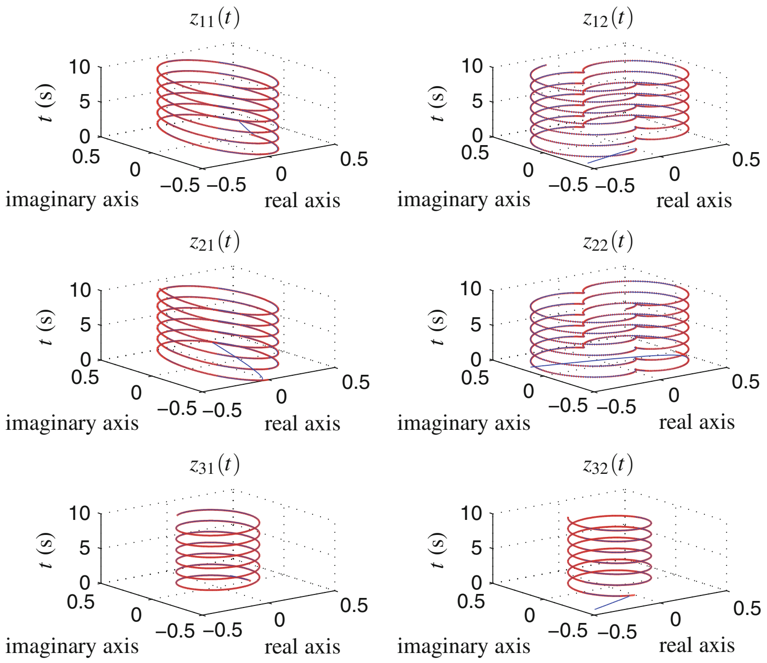


Fig. 13.6 State trajectories of complex ZD model (13.13) with $\gamma = 100$ and $\lambda = 10^{-3}$, where *dash-dotted curves* denote the theoretical time-varying pseudoinverse $C^{+}(t)$ in Example 13.2 and *solid curves* denote the solution computed by complex ZD model (13.13)

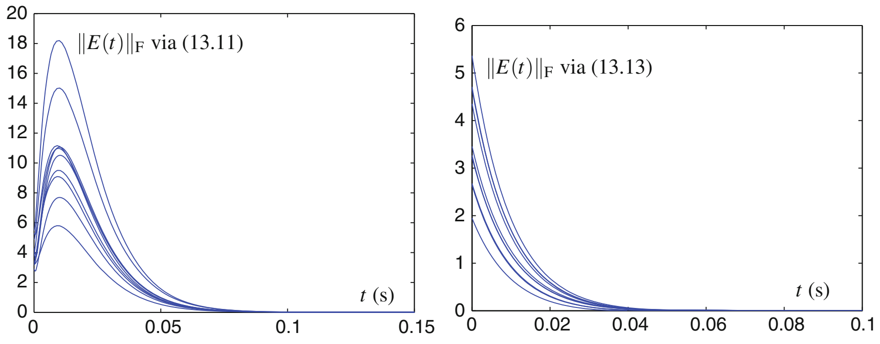


Fig. 13.7 Residual errors $\|E(t)\|_{\mathrm{F}} = \|Z(t)C(t)C^{\mathrm{H}}(t) - C^{\mathrm{H}}(t)\|_{\mathrm{F}}$ synthesized by complex ZD models (13.11) and (13.13) with $\gamma = 100$ and $\lambda = 10^{-3}$ for the time-varying pseudoinverse of matrix $C(t)$ in (13.19)

Example 13.3 In this example, we consider a more complicated situation of the time-varying complex generalized inverse (specifically, the pseudoinverse), which is the pseudoinverse of the following time-varying full-rank complex matrix:

$$C(t) = \begin{bmatrix} c_{11}(t) & c_{12}(t) & c_{13}(t) & \cdots & c_{1n}(t) \\ c_{21}(t) & c_{22}(t) & c_{23}(t) & \cdots & c_{2n}(t) \\ c_{31}(t) & c_{32}(t) & c_{33}(t) & \cdots & c_{3n}(t) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_{m1}(t) & c_{m2}(t) & c_{m3}(t) & \cdots & c_{mn}(t) \end{bmatrix} \in \mathbb{C}^{m\times n}, \tag{13.20}$$

where $m < n$. Thereinto,

$$c_{mn}(t) = \begin{cases} \exp(it), & \text{if } m = n, \\ n + \exp(-it), & \text{if } m > n, \\ m + \exp(-it), & \text{if } m < n. \end{cases}$$

In this example, due to the complexity of matrix $C(t)$ in (13.20) (with large dimensions, i.e., $m = 8$ and $n = 9$), the analytical theoretical pseudoinverse solution is difficult to be obtained. Therefore, we only present the convergence performance of the residual errors $\|E(t)\|_F = \|Z(t)C(t)C^H(t) - C^H(t)\|_F$ synthesized by complex ZD model (13.15). The simulation results are shown in Fig. 13.8. As seen from the figure, starting from 10 randomly-generated initial states, the residual errors $\|E(t)\|_F$ synthesized by complex ZD model (13.15) with $\gamma = 100$ can diminish to 0 within a short time (also about 0.06 s), which means that the corresponding solutions $Z(t)$ converge to the theoretical time-varying pseudoinverse of complex matrix matrix $C(t)$ in (13.20) rapidly and accurately. Thus, the efficacy of the proposed complex ZD model (13.15) on solving for the more complicated time-varying complex generalized

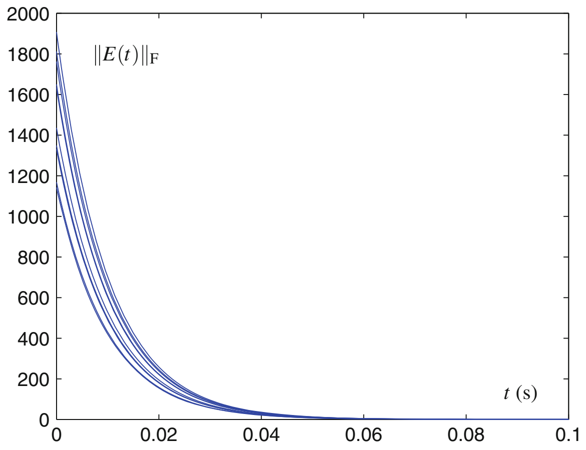


Fig. 13.8 Residual errors $\|E(t)\|_F = \|Z(t)C(t)C^H(t) - C^H(t)\|_F$ synthesized by complex ZD model (13.15) with $\gamma = 100$ for the time-varying pseudoinverse of matrix $C(t)$ in (13.20)

inverse (specifically, the pseudoinverse) is substantiated evidently. Note that, based on the above three examples, we can conclude that the convergence time of the proposed complex ZD models does not increase as the matrix dimension increases.

Example 13.4 In this example, we investigate the important effect of the design parameter γ for the convergent rate of the proposed complex ZD models. For illustrative purpose, we only exploit complex ZD model (13.17) to solve for the generalized inverse of complex matrix $C(t)$ in (13.19) (see also Example 13.2).

Note that, for complex ZD model (13.17), the initial state $Z(0)$ should be sufficiently close to $C^+(0)$. Moreover, $C^+(0)$ can be obtained from $C^+(t)$ presented in Example 13.2 by setting $t = 0$. As displayed in Fig. 13.9, we can clearly find that the residual errors $\|E(t)\|_\text{F} = \|C(t)Z(t) - I\|_\text{F}$ synthesized by complex ZD model (13.17) are decreasing faster as the value of design parameter γ increases (i.e., with $\gamma = 100, 200$ and 500). That is, with $\gamma = 100, 200$ and 500, the convergence time of the residual errors $\|E(t)\|_\text{F}$ diminishes from about 0.06 to 0.03 s, and even to 0.01 s. Note that the simulative results using other complex ZD models [i.e., (13.7), (13.11), (13.13) and (13.15)] are similar to those shown in Fig. 13.9, and are thus omitted due to results similarity. Being a topic of exercise, the corresponding simulative verifications of such four complex ZD models are left for interested readers. Thus, the efficacy of complex ZD model (13.17) is demonstrated. Meanwhile, we can draw the conclusion that the superior convergence performance of the proposed complex ZD models can be achieved by choosing a larger value of design parameter γ.

In summary, from the above four illustrative examples, we have substantiated the efficacy of the proposed complex ZD models (13.7), (13.11), (13.13), (13.15), and (13.17) on solving for time-varying complex generalized inverse (in most cases, the pseudoinverse). Besides, the important role of the design parameter γ in such complex ZD models has also been discussed and illustrated.

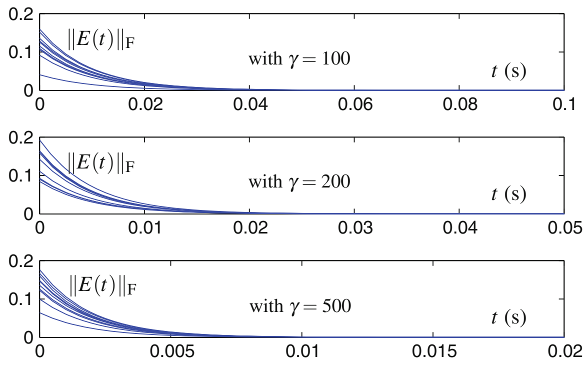


Fig. 13.9 Comparison on residual errors $\|E(t)\|_\text{F} = \|C(t)Z(t) - I\|_\text{F}$ synthesized by complex ZD model (13.17) with $\gamma = 100$, 200 and 500 for the time-varying pseudoinverse of matrix $C(t)$ in (13.19)

13.4 Summary

In this chapter, by defining different complex ZFs [i.e., (13.2)–(13.6)], five different complex ZD models [i.e., (13.7), (13.11), (13.13), (13.15), and (13.17)] have been proposed, generalized, developed and investigated for time-varying complex generalized inverse (in most cases, the pseudoinverse). Based on the complex ZF and the ZD design method, the complex ZD model has fully utilized the first-order time-derivative information of the time-varying complex matrix and has achieved the global convergence performance. In addition, the relationship between the proposed complex ZD models and the G-M dynamic system for time-varying complex generalized inverse computation has been discovered and presented. Moreover, through four illustrative examples, the efficacy of the proposed complex ZD models has been substantiated evidently.

References

1. Klein CA, Kee KB (1989) The nature of drift in pseudoinverse control of kinematically redundant manipulators. IEEE Trans Robot Autom 5(2):231–234
2. Feldkamp LA, Puskorius GV (1998) A signal processing framework based on dynamic neural networks with application to problem in adaptation, filtering, and classification. Proc IEEE 86(11):2259–2277
3. Zhang BL, Zhang H, Ge SS (2004) Face recognition by applying wavelet subband representation and kernel associative memory. IEEE Trans Neural Netw 15(1):166–177
4. Chountasis S, Katsikis VN, Pappas D (2010) Digital image reconstruction in the spectral domain utilizing the Moore-Penrose inverse. Math Prob Eng 2010:1–14
5. Chountasis S, Katsikis VN, Pappas D (2009) Applications of the Moore-Penrose inverse in digital image restoration. Math Prob Eng 2009:1–12
6. Courrieu P (2005) Fast computation of Moore-Penrose inverse matrices. Neural Inf Process Lett Rev 8(2):25–29
7. Perković MD, Stanimirović PS (2011) Iterative method for computing the Moore-Penrose inverse besed on Penrose equations. J Comput Appl Math 235(6):1604–1613
8. Fasano G (2007) Lanczos conjugate-gradient method and pseudoinverse computation on indefinite and singular systems. J Optim Theory Appl 132(2):267–285
9. Guo W, Huang T (2010) Method of elementary transformation to compute Moore-Penrose inverse. Appl Math Comput 216(5):1614–1617
10. Tasić MB, Stanimirović PS, Petković MD (2007) Symbolic computation of weighted Moore-Penrose inverse using partitioning method. Appl Math Comput 189(1):615–640
11. Zhang Y, Yi C (2011) Zhang neural networks and neural-dynamic method. Nova Science Publishers, New York
12. Guo D, Zhang Y (2012) Zhang neural network, Getz-Marsden dynamic system, and discrete-time algorithms for time-varying matrix inversion with application to robots' kinematic control. Neurocomputing 97:22–32
13. Hu J, Wang J (2012) Global stability of complex-valued recurrent neural networks with time-delays. IEEE Trans Neural Netw Learn Syst 23(6):853–865
14. Song J, Yam Y (1998) Complex recurrent neural network for computing the inverse and pseudo-inverse of the complex matrix. Appl Math Comput 93(2–3):195–205
15. Zhang Y, Li Z, Li K (2011) Complex-valued Zhang neural network for online complex-valued time-varying matrix inversion. Appl Math Comput 217(24):10066–10073

16. Ben-Israel A, Greville TNE (2003) Generalized inverses: theory and applications, 2nd edn. Springer, New York
17. Liao B, Zhang Y (2014) Different complex ZFs leading to different complex ZNN models for time-varying complex generalized inverse matrices. IEEE Trans Neural Netw Learn Syst 25(9):1621–1631
18. Rao CR, Mitra SK (1971) Generalized inverse of a matrix and its applications. Wiley, New York
19. Wang J (1997) Recurrent neural networks for computing pseudoinverses of rank-deficient matrices. SIAM J Sci Comput 19(5):1479–1493
20. Getz NH, Marsden JE (1997) Dynamical methods for polar decomposition and inversion of matrices. Linear Algebra Appl 258:311–343
21. Hopfield JJ (1982) Neural networks and physical systems with emergent collective computational abilities. Proc Natl Acad Sci USA 79(8):2554–2558
22. Mead C (1989) Analog VLSI and neural systems. Addison-Wesley Longman, Boston
23. Hjϕungnes A, Gesbert D (2007) Complex-valued matrix differentiation: techniques and key results. IEEE Trans Signal Process 55(6):2740–2746
24. Golub GH, Hansen PC, O'Leary DP (1999) Tikhonov regularization and total least squares. SIAM J Matrix Anal Appl 21(1):185–194
25. Zhang Y, Jiang D, Wang J (2002) A recurrent neural network for solving Sylvester equation with time-varying coefficients. IEEE Trans Neural Netw 13(5):1053–1063

Part V
ZF Application to Robot Control

Chapter 14
Application to Fixed-Base Robot RMP

Abstract In this chapter, the ZD approach presented in the previous chapters is applied to repetitive motion planning (RMP) of fixed-base redundant robot manipulators at the joint-acceleration level. Specifically, by introducing two different ZFs and by exploiting the ZD design formula, an acceleration-level RMP performance index is proposed, developed, and investigated. The resultant RMP scheme, which incorporates joint-angle, joint-velocity and joint-acceleration limits, is further presented and investigated to remedy the joint-angle drift phenomenon of fixed-base redundant robot manipulators. Such a scheme is then reformulated as a quadratic program, which is solved by a primal–dual neural network. With three path-tracking examples, simulation results based on PUMA560 robot manipulator substantiate well the effectiveness and accuracy of the proposed acceleration-level RMP scheme, as well as show the application prospect of the presented ZD approach.

14.1 Introduction

In recent years, robotic researchers have focused on solving a variety of tasks requiring sophisticated motion in complex environment via various advanced robots based on different planning and/or control methods [1–7]. Redundant robot manipulators are robots having more degrees of freedom (DOF) than required to perform a given end-effector primary task [6, 8, 9]. It has been argued that redundancy can improve the performance and versatility of a robot manipulator in various aspects such as obstacle avoidance [5, 10, 11], joint limits avoidance [6, 12, 13], and repetitive motion planning (RMP) [6, 9, 14]. Therefore, a multipurpose robot manipulator needs to be redundant if it is to be implemented effectively; e.g., a six-DOF PUMA560 robot manipulator has 3 redundant DOF when we consider only the end-effector's positioning, and it can thus perform various subtasks in addition to the end-effector's primary path-tracking task [6].

One fundamental issue on operating such a robot system is the redundancy-resolution problem [6]. The conventional solution to such a redundancy-resolution problem is the pseudoinverse-based method [15, 16]. The researches in recent years [5, 6, 8–10, 13] show that the redundancy-resolution problem can be solved in a more

© Springer-Verlag Berlin Heidelberg 2015
Y. Zhang and D. Guo, *Zhang Functions and Various Models*,
DOI 10.1007/978-3-662-47334-4_14

favorable manner via optimization techniques based on quadratic programming. Generally speaking, such optimization techniques are usually unified and expressed as a quadratic program (QP) which can incorporate equality, inequality, and bound constraints [5, 10]. The resultant QP problem can be solved by many methods and technics, such as dual neural network (DNN) [6, 8, 17] and primal–dual neural network (PDNN) [6, 9, 18]. Note that, in [19], comparisons between the DNN and the PDNN are demonstrated, which validates the latter has more advantages (e.g., PDNN is matrix-inversion free).

A redundancy-resolution scheme is called repetitive, if it maps closed paths in the task space (i.e., cyclic sequences of tasks) to closed trajectories in the configuration space (i.e., cyclic sequences of configurations) [6, 9, 14, 17, 18, 20, 21]. By contrast, the non-repetitive problem is that the joint angles may not return to their initial values when the end-effector traces a closed path in its workspace [6, 9, 14, 17, 18]. Note that the non-repetitive problem results in a joint angle drift phenomenon and may induce a problem that the manipulator's behavior is hard to predict [6, 9, 17, 18]; and it is then less efficient to readjust the manipulator's configuration after every cycle via self-motion [6, 9, 17, 18].

The previous researches on solving the non-repetitive problem are mainly at the joint-velocity level [6, 14, 17, 18, 20, 21]. However, these may not be applicable to the manipulators which are controlled at the acceleration and/or torque levels. In addition, the joint-acceleration physical limits of the manipulators cannot be incorporated in the scheme resolved at the joint-velocity level [6, 17, 18]. Thus, the RMP performance index at the joint-acceleration level is a very appealing and interesting topic in robotics research domain. Moreover, the acceleration-level scheme can effectively prevent the instability/divergence problem of joint accelerations and torques caused by some velocity-level scheme in long-range motion [9, 22].

In this chapter, based on the ZD approach presented in the previous chapters, we propose and investigate a novel RMP scheme at the joint-acceleration level to remedy the joint-angle drift phenomenon. Specifically, by introducing two different ZFs and by exploiting the ZD design formula [23], an acceleration-level RMP performance index is proposed, developed, and investigated. To the best of the authors' knowledge, such a new RMP performance index at the joint-acceleration level has never been investigated before by others. In addition, the proposed acceleration-level RMP scheme, which incorporates joint-angle, joint-velocity and joint-acceleration limits, is reformulated as a QP, and is then solved by a PDNN [6, 9, 18]. Moreover, simulation results based on PUMA560 robot manipulator performing different types of end-effector path-tracking tasks substantiate well the effectiveness and accuracy of the proposed acceleration-level RMP scheme, as well as show the application prospect of the presented ZD approach. Besides, it is worth pointing out here that our previous book [6] only presents and investigates RMP of redundant robot manipulators at the joint-velocity level. By contrast, in this book (or specifically, in this chapter), we focus on the investigation of RMP at the joint-acceleration level. Evidently, this is a great contribution and improvement, as it promotes the RMP research from joint-velocity level to joint-acceleration level.

14.2 RMP Performance Index Derived via Different ZFs

In this section, the ZD approach presented in the previous chapters is applied to deriving the RMP performance index at the joint-acceleration level.

To lay a basis for further discussion, some well-known essential equations for redundant robot manipulators can be given below directly [9, 15, 16]:

$$f(\theta) = \mathbf{r}, \tag{14.1}$$

$$J(\theta)\dot{\theta} = \dot{\mathbf{r}}, \tag{14.2}$$

$$J(\theta)\ddot{\theta} = \ddot{\mathbf{r}} - \dot{J}(\theta)\dot{\theta}. \tag{14.3}$$

For the above equations, (14.1) describes the relationship between the end-effector position-and-orientation vector $\mathbf{r} \in \mathbb{R}^m$ and joint-angle vector $\theta \in \mathbb{R}^n$, where $f(\cdot)$ is a differentiable nonlinear function with a structure and parameters which are known for a given robot manipulator. $\dot{\mathbf{r}}$ and $\dot{\theta}$ in (14.2) [by differentiating (14.1) with respect to time t] denote respectively the end-effector velocity vector and the joint-velocity vector, and $J(\theta) \in \mathbb{R}^{m\times n}$ is the Jacobian matrix defined as $J(\theta) = \partial f(\theta)/\partial\theta$. In (14.3) [by differentiating (14.2)], $\ddot{\mathbf{r}}$ and $\ddot{\theta}$ denote, respectively, the end-effector acceleration vector and the joint-acceleration vector, and $\dot{J}(\theta)$ is the time derivative of Jacobian matrix $J(\theta)$. For redundant robot manipulators (i.e., $m < n$), (14.1)–(14.3) are all underdetermined and generally admit an infinite number of solutions in terms of inverse kinematics. Besides, it is worth mentioning here that the RMP schemes investigated in [6, 17, 18] are based on (14.2) (i.e., at the joint-velocity level), whereas the RMP scheme presented in this chapter is based on (14.3) (i.e., at the joint-acceleration level).

Now we present the specific design procedure to obtain the acceleration-level RMP performance index as follows:

- Firstly, to achieve RMP of both end-effector and joints, we require that $r(T_\mathrm{d}) = r(0)$ and $\theta(T_\mathrm{d}) = \theta(0)$, where T_d denotes the task duration of both path-tracking and RMP, and $\theta(0)$ denotes the initial value of $\theta(t)$ [i.e., $\theta(0) = \theta(t = 0)$]. Thus, by following the ZD approach, it is natural to define the following vector-valued joint-displacement function (i.e., the first ZF):

$$\mathbf{e}(t) = \theta(t) - \theta(0) \in \mathbb{R}^n.$$

- Secondly, to eliminate every entry of the vector-valued joint-displacement function $\mathbf{e}(t)$ over $[0, T_\mathrm{d}]$, we exploit the ZD design formula [i.e., (4.2)] as follows:

$$\dot{\mathbf{e}}(t) = -\gamma\mathbf{e}(t) = -\gamma(\theta(t) - \theta(0)) \tag{14.4}$$

where design parameter $\gamma > 0 \in \mathbb{R}$ is used to adjust the exponential convergence rate of $\mathbf{e}(t)$ to zero. Since the time derivative of $\mathbf{e}(t)$ is $\dot{\mathbf{e}}(t) = \dot{\theta}(t)$, (14.4) is rewritten as

$$\dot{\theta}(t) + \gamma(\theta(t) - \theta(0)) = \mathbf{0} \in \mathbb{R}^n. \tag{14.5}$$

- Thirdly, to achieve (14.5), we define the second ZF as follows:

$$\mathbf{e}(t) = \dot{\theta}(t) + \gamma(\theta(t) - \theta(0)) \in \mathbb{R}^n.$$

  By exploiting the ZD design formula (4.2) again, we have $\ddot{\theta}(t) + \gamma\dot{\theta}(t) = -\gamma(\dot{\theta}(t) + \gamma(\theta(t) - \theta(0)))$, which is reformulated as

$$\ddot{\theta}(t) + 2\gamma\dot{\theta}(t) + \gamma^2(\theta(t) - \theta(0)) = \mathbf{0} \in \mathbb{R}^n.$$

- Finally, as the end-effector's task requirement and joint physical limits should be considered, it is better to minimize the performance index $\|\ddot{\theta}(t) + 2\gamma\dot{\theta}(t) + \gamma^2(\theta(t) - \theta(0))\|_2^2/2$, rather than using $\ddot{\theta}(t) + 2\gamma\dot{\theta}(t) + \gamma^2(\theta(t) - \theta(0)) = \mathbf{0}$ directly. Therefore, with $\mathbf{h} = 2\gamma\dot{\theta}(t) + \gamma^2(\theta(t) - \theta(0))$ defined, we obtain

$$\|\ddot{\theta}(t) + 2\gamma\dot{\theta}(t) + \gamma^2(\theta(t) - \theta(0))\|_2^2/2 = \|\ddot{\theta}(t) + \mathbf{h}\|_2^2/2 = (\ddot{\theta} + \mathbf{h})^{\mathrm{T}}(\ddot{\theta} + \mathbf{h})/2, \tag{14.6}$$

  which is the acceleration-level RMP performance index for fixed-base redundant robot manipulators.

In summary, by using the ZD approach, we have developed the RMP performance index (14.6) at the joint-acceleration level (showing the application prospect of such a ZD approach). Note that, by minimizing the acceleration-level performance index (14.6) [i.e., "minimize $(\ddot{\theta} + \mathbf{h})^{\mathrm{T}}(\ddot{\theta} + \mathbf{h})/2$"], the RMP purpose is thus achieved for fixed-base redundant robot manipulators.

14.3 Scheme and QP Formulations

In this section, based on the proposed performance index (14.6), a novel RMP scheme is further developed and investigated for fixed-base redundant robot manipulators at the joint-acceleration level. In addition, such an acceleration-level RMP scheme is reformulated as a QP, which is solved by a primal–dual neural network [6, 9, 18, 19].

14.3.1 Acceleration-Level RMP Scheme

With joint physical limits (i.e., joint-angle, joint-velocity, and joint-acceleration limits) being considered, the acceleration-level RMP scheme is proposed as follows:

$$\begin{aligned}
\text{minimize} \quad & (\ddot{\theta}+\mathbf{h})^{\mathrm{T}}(\ddot{\theta}+\mathbf{h})/2 && (14.7)\\
\text{subject to} \quad & J(\theta)\ddot{\theta}=\ddot{\mathbf{r}}_{\mathrm{d}}-\dot{J}(\theta)\dot{\theta}, && (14.8)\\
& \theta^{-}\leqslant\theta\leqslant\theta^{+}, && (14.9)\\
& \dot{\theta}^{-}\leqslant\dot{\theta}\leqslant\dot{\theta}^{+}, && (14.10)\\
& \ddot{\theta}^{-}\leqslant\ddot{\theta}\leqslant\ddot{\theta}^{+}, && (14.11)
\end{aligned}$$

where $\ddot{\mathbf{r}}_{\mathrm{d}}$ denotes the twice time derivative of the desired end-effector path $\mathbf{r}_{\mathrm{d}}\in\mathbb{R}^m$. In addition, $\theta^{\pm}$, $\dot{\theta}^{\pm}$ and $\ddot{\theta}^{\pm}$ denote the upper and lower limits of the joint-angle, joint-velocity, and joint-acceleration vectors, respectively.

14.3.2 Bound Constraint Transformation Technique

Since the proposed RMP scheme (14.7)–(14.11) is resolved at the joint-acceleration level, the constraints in (14.9)–(14.11) have to be converted to the expressions in terms of joint acceleration $\ddot{\theta}$. In view of the inertia movement, the avoidance of the upper limit of the ith joint (i.e., θ_i^{+}) in (14.9) can be converted as

$$\ddot{\theta}_i\leqslant\kappa_\alpha(\lambda\theta_i^{+}-\theta_i), \tag{14.12}$$

and the avoidance of the lower limit of the ith joint (i.e., θ_i^{-}) in (14.9) can be converted as

$$\ddot{\theta}_i\geqslant\begin{cases}\kappa_\alpha(\lambda\theta_i^{-}-\theta_i), & \text{for } \theta_i^{-}<0\\ \kappa_\alpha(\theta_i^{-}+\vartheta-\theta_i), & \text{for } \theta_i^{-}\geqslant 0\end{cases} \tag{14.13}$$

where design parameters $\lambda\in(0,1)$ and $\vartheta>0\in\mathbb{R}$ are selected (e.g., $\lambda=0.9$ and $\vartheta=0.0524$ rad), to define critical regions $[\theta_i^{-},\lambda\theta_i^{-}]$ or $[\theta_i^{-},\theta_i^{-}+\vartheta]$ and $[\lambda\theta_i^{+},\theta_i^{+}]$ for joint position variables such that there will appear a deceleration when the robot manipulator enters them [9, 22]. In addition, $\kappa_\alpha>0\in\mathbb{R}$ determines the magnitude of such a deceleration. Similarly, the avoidance of the ith joint-velocity limits $\dot{\theta}_i^{\pm}$ in (14.10) can be converted as

$$\kappa_\beta(\dot{\theta}_i^{-}-\dot{\theta}_i)\leqslant\ddot{\theta}_i\leqslant\kappa_\beta(\dot{\theta}_i^{+}-\dot{\theta}_i), \tag{14.14}$$

which guarantees that joint acceleration changes its direction gradually as the joint velocity approaches its limit. Design parameters κ_α and κ_β are selected such that the feasible region of $\ddot{\theta}$ made by the conversion of joint-angle limits and joint-velocity limits [i.e., (14.9) and (14.10)] are normally not smaller than the original one made by joint-acceleration limits, i.e., bound constraint (14.11). By using (14.12)–(14.14), the acceleration-level avoidance of joint physical limits (14.9)–(14.11) becomes $\zeta^- \leqslant \ddot{\theta} \leqslant \zeta^+$. Here, ζ^- and ζ^+ denote, respectively, the resultant lower bound and upper bound synthesized by the joint-angle limits, joint-velocity limits, and joint-acceleration limits. In addition, the ith elements of ζ^- and ζ^+ are defined respectively as

$$\zeta_i^- = \begin{cases} \max\{\kappa_\alpha(\lambda\theta_i^- - \theta_i), \kappa_\beta(\dot{\theta}_i^- - \dot{\theta}_i), \ddot{\theta}_i^-\}, & \text{for } \theta_i^- < 0 \\ \max\{\kappa_\alpha(\theta_i^- + \vartheta - \theta_i), \kappa_\beta(\dot{\theta}_i^- - \dot{\theta}_i), \ddot{\theta}_i^-\}, & \text{for } \theta_i^- \geqslant 0 \end{cases}$$
$$\zeta_i^+ = \min\{\kappa_\alpha(\lambda\theta_i^+ - \theta_i), \kappa_\beta(\dot{\theta}_i^+ - \dot{\theta}_i), \ddot{\theta}_i^+\}.$$

14.3.3 QP Reformulation

In (14.7), $(\ddot{\theta} + \mathbf{h})^{\mathrm{T}}(\ddot{\theta} + \mathbf{h})/2 = (\ddot{\theta}^{\mathrm{T}}\ddot{\theta} + \ddot{\theta}^{\mathrm{T}}\mathbf{h} + \mathbf{h}^{\mathrm{T}}\ddot{\theta} + \mathbf{h}^{\mathrm{T}}\mathbf{h})/2$. Since the proposed RMP scheme is resolved at the joint-acceleration level and the decision variable vector is joint acceleration $\ddot{\theta}$, the parameter $\mathbf{h}$ [i.e., $\mathbf{h} = 2\gamma\dot{\theta} + \gamma^2(\theta - \theta(0))$ in (14.7)] is viewed as a constant in the performance index. In this situation, $\mathbf{h}^{\mathrm{T}}\mathbf{h}/2$ is also viewed as a constant (with respect to $\ddot{\theta}$) and $\mathbf{h}^{\mathrm{T}}\mathbf{h}/2$ is thus set aside from the performance index. Therefore, the minimization of (14.7) is equivalent to the minimization of $\ddot{\theta}^{\mathrm{T}}\ddot{\theta}/2 + \mathbf{h}^{\mathrm{T}}\ddot{\theta}$ (note that $\ddot{\theta}^{\mathrm{T}}\mathbf{h} = \mathbf{h}^{\mathrm{T}}\ddot{\theta}$).

In light of the above minimization formula and the above bound constraint conversion, with $\mathbf{x} = \ddot{\theta} \in \mathbb{R}^n$ and $W = I \in \mathbb{R}^{n\times n}$, the proposed acceleration-level RMP scheme (14.7)–(14.11) for physically-constrained redundant robot manipulators is reformulated finally as the following QP:

$$\text{minimize} \quad \mathbf{x}^{\mathrm{T}}W\mathbf{x}/2 + \mathbf{h}^{\mathrm{T}}\mathbf{x} \tag{14.15}$$
$$\text{subject to} \quad C\mathbf{x} = \mathbf{d}, \tag{14.16}$$
$$\zeta^- \leqslant \mathbf{x} \leqslant \zeta^+, \tag{14.17}$$

where $C = J(\theta)$ and $\mathbf{d} = \ddot{\mathbf{r}}_{\mathrm{d}} - \dot{J}(\theta)\dot{\theta}$.

14.3.4 QP Solver

According to [6, 9, 18, 19], the presented QP problem (14.15)–(14.17) can be solved by the following primal–dual neural network (PDNN):

$$\dot{\mathbf{u}} = \nu(I + M^{\mathrm{T}})\{P_\Omega(\mathbf{u} - (M\mathbf{u} + \mathbf{p})) - \mathbf{u}\} \tag{14.18}$$

where design parameter $\nu > 0 \in \mathbb{R}$ is used to scale the convergence rate of the neural network. The piecewise-linear activation-function array $P_{\Omega}(\cdot)$ can be implemented by using operational amplifiers. In addition, $\Omega = \{\mathbf{u} \in \mathbb{R}^{n+m} | \mathbf{u}^- \leqslant \mathbf{u} \leqslant \mathbf{u}^+\} \subset \mathbb{R}^{n+m}$,

$$\mathbf{u} = \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \in \mathbb{R}^{n+m},\ \mathbf{u}^- = \begin{bmatrix} \zeta^- \\ -\varpi \mathbf{1}_{\mathrm{v}} \end{bmatrix} \in \mathbb{R}^{n+m},\ \mathbf{u}^+ = \begin{bmatrix} \zeta^+ \\ \varpi \mathbf{1}_{\mathrm{v}} \end{bmatrix} \in \mathbb{R}^{n+m},$$

$$M = \begin{bmatrix} W & -C^{\mathrm{T}} \\ C & 0 \end{bmatrix} \in \mathbb{R}^{(n+m)\times(n+m)},\ \mathbf{p} = \begin{bmatrix} \mathbf{h} \\ -\mathbf{d} \end{bmatrix} \in \mathbb{R}^{n+m},\ \mathbf{1}_{\mathrm{v}} = [1, \ldots, 1]^{\mathrm{T}} \in \mathbb{R}^{m}.$$

It is worth mentioning that $P_{\Omega}(\cdot) : \mathbb{R}^{n+m} \to \Omega$ is a projection operator, with the ith element of $P_{\Omega}(\mathbf{u})$ defined as

$$\begin{cases} u_i^-, & \text{if } u_i < u_i^-, \\ u_i, & \text{if } u_i^- \leqslant u_i \leqslant u_i^+,\ \forall i \in \{1, 2, 3, \ldots, n+m\}. \\ u_i^+, & \text{if } u_i > u_i^+, \end{cases}$$

$\mathbf{y} \in \mathbb{R}^m$ is the dual decision vector defined for equality constraint (14.16), and $\varpi \gg 0$ is defined sufficiently large (e.g., $\varpi = 10^6$) to replace $+\infty$ numerically. Furthermore, we have the following theorem which guarantees that PDNN (14.18) can globally generate optimal solution x^* to QP (14.15)–(14.17) [6, 9, 18, 19].

Theorem 14.1 *Assume the existence of optimal solution* $\mathbf{x}^*$ *to QP (14.15)–(14.17). Starting from any initial state* $\mathbf{u}(0)$*, state vector* $\mathbf{u}(t)$ *of PDNN (14.18) converges to equilibrium point* $\mathbf{u}^*$*, of which the first n elements constitute the optimal solution* $\mathbf{x}^*$ *to QP (14.15)–(14.17).*

14.4 Illustrative Examples

In this section, to substantiate the efficacy of the proposed acceleration-level RMP scheme (14.7)–(14.11), computer simulations are performed based on PUMA560 robot manipulator, of which the mechanical configuration is shown in Fig. 14.1 and the joint physical parameters (i.e., joint-angle limits $\theta^{\pm}$, joint-velocity limits $\dot{\theta}^{\pm}$, joint-acceleration limits $\ddot{\theta}^{\pm}$) used in the ensuing simulations are given in Table 14.1. In the simulations, the end-effector of the PUMA560 robot manipulator is required to track three different paths, i.e., a pentagram path, an East-Asian character looking like symbol "⊟" which means "the sun" in Chinese, and the initial letter "V" of the English word "VICTORY." Note that, when we apply PDNN (14.18) to solving

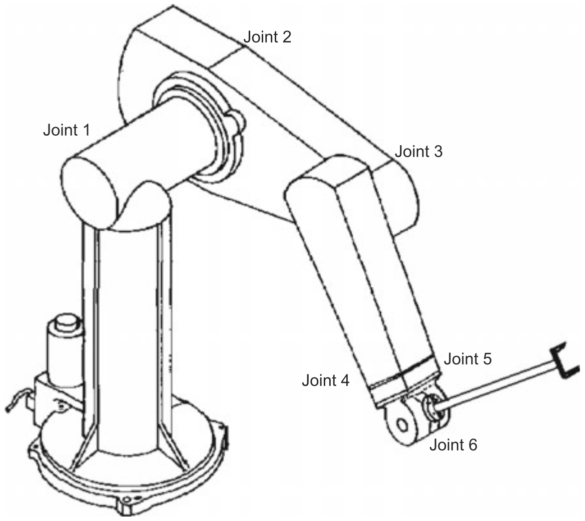


Fig. 14.1 Mechanical configuration of PUMA560 robot manipulator used in simulations

Table 14.1 Joint physical limits used in the PUMA560 simulations

#	θ^+ (rad)	θ^- (rad)	$\dot{\theta}^+$ (rad/s)	$\dot{\theta}^-$ (rad/s)	$\ddot{\theta}^+$ (rad/s^2)	$\ddot{\theta}^-$ (rad/s^2)
1	+2.775	−2.775	+1.5	−1.5	+6.0	−6.0
2	+0.750	−3.892	+1.5	−1.5	+6.0	−6.0
3	+4.049	−0.905	+1.5	−1.5	+6.0	−6.0
4	+2.967	−1.919	+1.5	−1.5	+6.0	−6.0
5	+1.745	−1.745	+1.5	−1.5	+6.0	−6.0
6	+4.625	−4.625	+1.5	−1.5	+6.0	−6.0

the presented QP problem (14.15)–(14.17) [as well as the proposed acceleration-level RMP scheme (14.7)–(14.11)] for controlling the PUMA560 robot manipulator, design parameter $\nu = 10^5$ is used throughout this chapter.

14.4.1 Pentagram-Path Tracking

In the computer simulations of this subsection, the PUMA560 robot manipulator's end-effector is expected to move along a pentagram path with the side length being 0.0707 m. The X-axis, Y-axis, and Z-axis acceleration functions of the desired pentagram path are

$$\ddot{r}_{\mathrm{X}}(t)=\begin{cases}-\frac{\sqrt{2}l\pi}{T^2}\cos(\frac{\pi}{5})\sin(\frac{2\pi t}{T}), & \forall t\in[0,T]\\ -\frac{\sqrt{2}l\pi}{T^2}\cos(\frac{\pi}{5})\sin[\frac{2\pi(t-T)}{T}], & \forall t\in[T,2T]\\ \frac{\sqrt{2}l\pi}{T^2}\cos(\frac{2\pi}{5})\sin[\frac{2\pi(t-2T)}{T}], & \forall t\in[2T,3T]\\ -\frac{\sqrt{2}l\pi}{T^2}\cos(\frac{\pi}{5})\sin[\frac{2\pi(t-3T)}{T}], & \forall t\in[3T,4T]\\ \frac{\sqrt{2}l\pi}{T^2}\sin[\frac{2\pi(t-4T)}{T}], & \forall t\in[4T,5T]\\ \frac{\sqrt{2}l\pi}{T^2}\sin(\frac{\pi}{10})\sin[\frac{2\pi(t-5T)}{T}], & \forall t\in[5T,6T]\\ \frac{\sqrt{2}l\pi}{T^2}\sin(\frac{\pi}{10})\sin[\frac{2\pi(t-6T)}{T}], & \forall t\in[6T,7T]\\ \frac{\sqrt{2}l\pi}{T^2}\sin[\frac{2\pi(t-7T)}{T}], & \forall t\in[7T,8T]\\ -\frac{\sqrt{2}l\pi}{T^2}\cos(\frac{\pi}{5})\sin[\frac{2\pi(t-8T)}{T}], & \forall t\in[8T,9T]\\ \frac{\sqrt{2}l\pi}{T^2}\cos(\frac{2\pi}{5})\sin[\frac{2\pi(t-9T)}{T}], & \forall t\in[9T,10T]\end{cases}$$

$$\ddot{r}_{\mathrm{Y}}(t)=\begin{cases}-\frac{\sqrt{2}l\pi}{T^2}\sin(\frac{\pi}{5})\sin(\frac{2\pi t}{T}), & \forall t\in[0,T]\\ -\frac{\sqrt{2}l\pi}{T^2}\sin(\frac{\pi}{5})\sin[\frac{2\pi(t-T)}{T}], & \forall t\in[T,2T]\\ \frac{\sqrt{2}l\pi}{T^2}\sin(\frac{2\pi}{5})\sin[\frac{2\pi(t-2T)}{T}], & \forall t\in[2T,3T]\\ \frac{\sqrt{2}l\pi}{T^2}\sin(\frac{\pi}{5})\sin[\frac{2\pi(t-3T)}{T}], & \forall t\in[3T,4T]\\ 0, & \forall t\in[4T,5T]\\ \frac{\sqrt{2}l\pi}{T^2}\cos(\frac{\pi}{10})\sin[\frac{2\pi(t-5T)}{T}], & \forall t\in[5T,6T]\\ -\frac{\sqrt{2}l\pi}{T^2}\cos(\frac{\pi}{10})\sin[\frac{2\pi(t-6T)}{T}], & \forall t\in[6T,7T]\\ 0, & \forall t\in[7T,8T]\\ -\frac{\sqrt{2}l\pi}{T^2}\sin(\frac{\pi}{5})\sin[\frac{2\pi(t-8T)}{T}], & \forall t\in[8T,9T]\\ -\frac{\sqrt{2}l\pi}{T^2}\sin(\frac{2\pi}{5})\sin[\frac{2\pi(t-9T)}{T}], & \forall t\in[9T,10T]\end{cases}$$

$$\ddot{r}_{\mathrm{Z}}(t)=0,\forall t\in[0,10T]$$

where the task duration T_{d} is $10T$, and parameter l should be set appropriately according to the desired length of the line segment in a path-tracking task. Specifically, in the computer simulation of this subsection, $l = 0.1\,\mathrm{m}$ and the task duration T_{d} is $3.9\,\mathrm{s}$ (i.e., $T = 0.39\,\mathrm{s}$). In addition, during such a task execution, the joints of PUMA560 robot manipulator are expected to start from initial state $\theta(0) = [0, -\pi/4, 0, \pi/2, -\pi/4, 0]^{\mathrm{T}}$ rad, and finally return to the initial state.

First, the inverse kinematics problem of PUMA560 robot manipulator is handled via QP (14.15)–(14.17) with neither joint physical limits nor RMP criterion considered. That is, $\theta^{\pm}$ in (14.9), $\dot{\theta}^{\pm}$ in (14.10) and $\ddot{\theta}^{\pm}$ in (14.11) are set as $\pm\varpi 1_{\mathrm{v}}$, and in (14.7), the RMP coefficient $\gamma = 0$. The corresponding simulation results are shown in Fig. 14.2. It is seen from the upper graph of Fig. 14.2 that the end-effector's trajectory is the desired pentagram curve, which shows that the end-effector's primary task

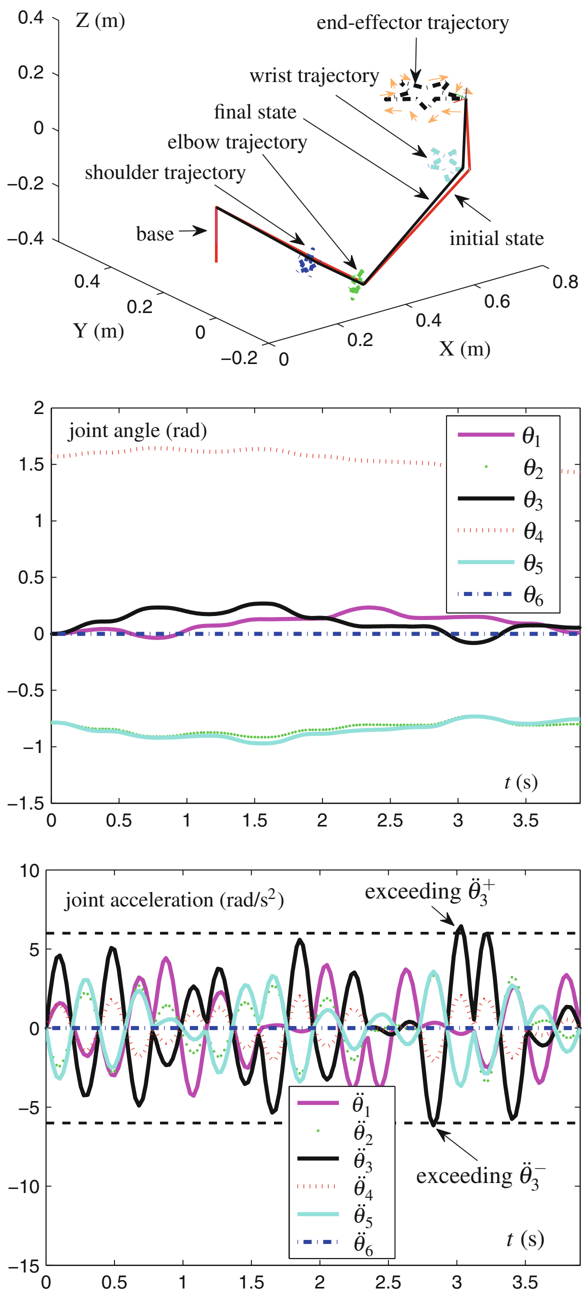


Fig. 14.2 PUMA560 end-effector tracks a pentagram path with neither joint physical limits nor RMP criterion considered

is completed. However, the final state of the robot manipulator (denoted in black) has not returned to the initial one (denoted in red). This can also be seen from the middle graph of Fig. 14.2 which illustrates the θ profiles over the task duration. The exact actual values of the final joints, $\theta(3.9)$, are shown in the second column of Table 14.2, which are evidently different from the initial ones, $\theta(0)$, in the third column. That is to say, the joint angle drift phenomenon has happened. It is worth mentioning that this phenomenon is not desired in industrial applications because we need extra self-motion to readjust the manipulator's configuration. This would thus lead to low efficiency. Besides, the θ and $\ddot{\theta}$ profiles during the task execution are shown, respectively, in the middle and lower graphs of Fig. 14.2. Compared with the joint physical limits in Table 14.1, the joint angles and joint velocities in the simulations do not reach their corresponding limits; i.e., the joint-angle limits and joint-velocity limits have not been activated. However, joint acceleration $\ddot{\theta}_3(t)$ exceeds its lower limit -6 rad/s^2 at about $t = 2.83$ s and its upper limit $+6$ rad/s^2 at about $t = 3.03$ s, which may lead to the damage to the manipulator and is less desirable in applications.

Second, for comparison and for illustration, the simulation results with both joint physical limits and RMP criterion considered are shown in Fig. 14.3. That is, $\theta^{\pm}$, $\dot{\theta}^{\pm}$, and $\ddot{\theta}^{\pm}$ are set correctly according to Table 14.1, $\lambda = 0.9$, $\kappa_\alpha = \kappa_\beta = 20$, and $\gamma = 6$. As seen from the upper left graph of Fig. 14.3, the end-effector of PUMA560 moves along a pentagram path, which is sufficiently close to the desired one. In addition, this solution is repetitive and applicable for PUMA560 working with a cyclic pentagram-path tracking requirement, because the final and initial states coincide well with each other (in the upper right graph of Fig. 14.3). Furthermore, as seen from Fig. 14.3 and Table 14.3, all joint variables are kept within their limited ranges. Besides, the lower left graph of Fig. 14.3 shows the motion trajectories of PUMA560 over the task duration. The end-effector positioning error $\varepsilon = r_\mathrm{d} - f(\theta)$ is shown in the lower right graph of Fig. 14.3, illustrating small deviations of the end-effector from the desired path r_d in the X-axis, Y-axis, and Z-axis of the base frame (i.e., ε_X, ε_Y and ε_Z, respectively). As seen from the figure, the maximal component of the positioning error is less than 5.0×10^{-4} m, which illustrates the accuracy of the PUMA560 robot

Table 14.2 Joint drifts (rad) with neither joint limits nor RMP criterion considered when PUMA560 end-effector tracks the pentagram path

Joint	$\theta(3.9)$	$\theta(0)$	$\theta(3.9) - \theta(0)$
θ_1	+0.00957950482	+0.00000000000	+0.00957950482
θ_2	−0.80021692574	−0.78539816340	−0.01481876234
θ_3	+0.05479385946	+0.00000000000	+0.05479385946
θ_4	+1.42816130434	+1.57079632680	−0.14263502246
θ_5	−0.75586229472	−0.78539816340	+0.02953586868
θ_6	+0.00000000000	+0.00000000000	+0.00000000000

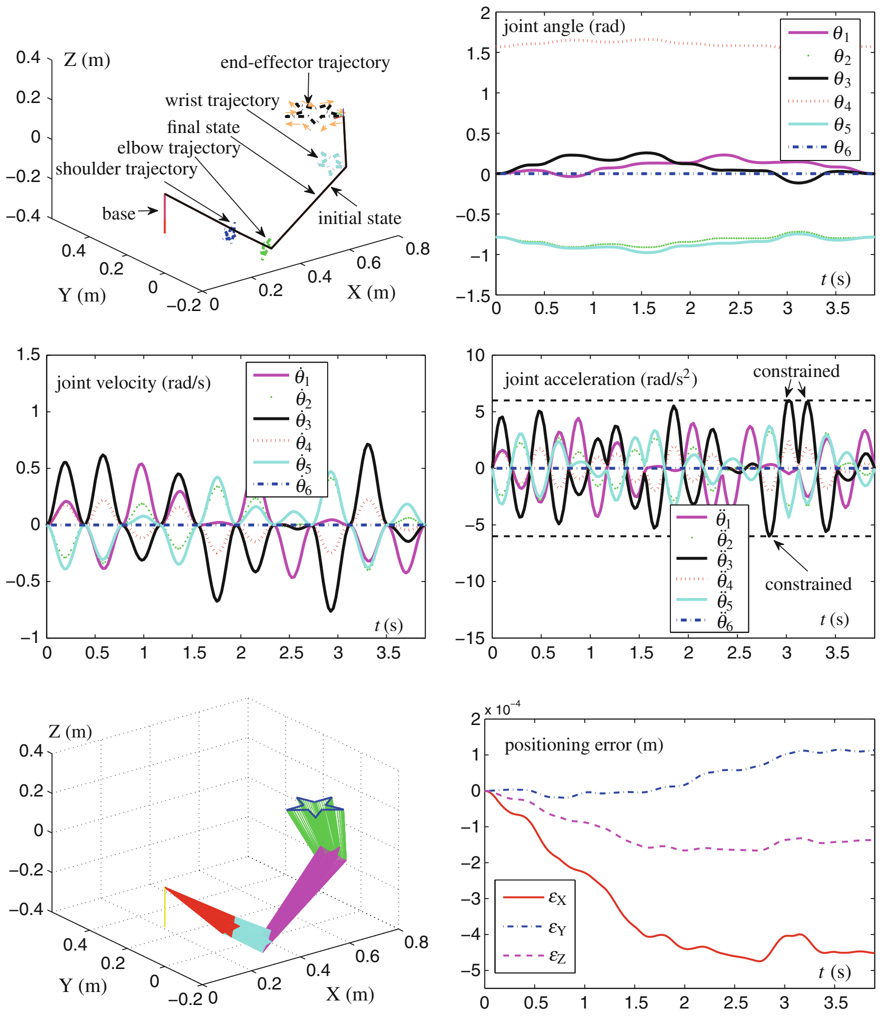


Fig. 14.3 PUMA560 end-effector tracks the pentagram path with both joint physical limits and RMP criterion considered, where PUMA560 final state coincides with its initial state

manipulator synthesized by the proposed acceleration-level RMP scheme when the end-effector tracks the desired pentagram path.

Third, comparing Tables 14.2 and 14.3, we can see that the joint angle drifts in the former (i.e., Table 14.2) are large; in contrast, the joint angle drifts in the latter (i.e., Table 14.3) are quite small (less than 3.41×10^{-3} rad). This shows quantificationally the repetitive accuracy of the proposed RMP scheme. In summary, this example has substantiated well the efficacy of the proposed acceleration-level RMP scheme (14.7)–(14.11) and its QP solver (14.18) (i.e., a PDNN) on redundancy resolution

Table 14.3 Joint drifts (rad) with both joint limits and RMP criterion considered when PUMA560 end-effector tracks a pentagram path

Joint	$\theta(3.9)$	$\theta(0)$	$\theta(3.9) - \theta(0)$
θ_1	$+0.00000629838$	$+0.00000000000$	$+6.29838 \times 10^{-6}$
θ_2	-0.78469374929	-0.78539816340	$+7.04414 \times 10^{-4}$
θ_3	-0.00096763343	$+0.00000000000$	-9.67633×10^{-4}
θ_4	$+1.57045580322$	$+1.57079632680$	-3.40524×10^{-3}
θ_5	-0.78474817868	-0.78539816340	$+6.49985 \times 10^{-4}$
θ_6	$+0.00000000000$	$+0.00000000000$	$+0.00000000000$

of physically-constrained robot manipulators, and more importantly, has shown the application prospect of the presented ZD approach.

14.4.2 East-Asian Character Writing

In the simulations of this subsection, the motion trajectory of the PUMA560 end-effector is expected to be an East-Asian character looking like symbol "⊟" which means "the sun" in Chinese. The word width is 0.1697 m and the word height is 0.3394 m. The X-axis, Y-axis, and Z-axis acceleration functions of the path are

$$\ddot{r}_{\mathrm{X}}(t) = \begin{cases} -\frac{\sqrt{2}\iota\pi}{T^2}\sin(\frac{2\pi t}{T}), & \forall t \in [0, T] \\ 0, & \forall t \in [T, 2T] \\ \frac{\sqrt{2}\iota\pi}{T^2}\sin[\frac{2\pi(t-2T)}{T}], & \forall t \in [2T, 3T] \\ 0, & \forall t \in [3T, 4T] \\ -\frac{\sqrt{2}\iota\pi}{T^2}\sin[\frac{2\pi(t-4T)}{T}], & \forall t \in [4T, 5T] \\ 0, & \forall t \in [5T, 6T] \\ \frac{\sqrt{2}\iota\pi}{T^2}\sin[\frac{2\pi(t-6T)}{T}], & \forall t \in [6T, 7T] \end{cases}$$

$$\ddot{r}_{\mathrm{Y}}(t) = \begin{cases} 0, & \forall t \in [0, T] \\ \frac{\sqrt{2}\iota\pi}{T^2}\sin[\frac{2\pi(t-T)}{T}], & \forall t \in [T, 2T] \\ 0, & \forall t \in [2T, 3T] \\ -\frac{2\sqrt{2}\iota\pi}{T^2}\sin[\frac{2\pi(t-3T)}{T}], & \forall t \in [3T, 4T] \\ 0, & \forall t \in [4T, 5T] \\ \frac{\sqrt{2}\iota\pi}{T^2}\sin[\frac{2\pi(t-5T)}{T}], & \forall t \in [5T, 6T] \\ 0, & \forall t \in [6T, 7T] \end{cases}$$

$$\ddot{r}_{\mathrm{Z}}(t) = 0, \forall t \in [0, 7T]$$

where the task duration T_d is $7T$, and parameter ι should be set appropriately according to the desired length of the line segment in a path-tracking task. Specifically, in the simulations of this subsection, design parameter $\iota = 0.24\,\mathrm{m}$ and the task duration T_d is 4.55 s (i.e., $T = 0.65\,\mathrm{s}$). In addition, the initial joint state $\theta(0) = [0, -\pi/4, 0, \pi/2, -\pi/4, 0]^\mathrm{T}$ rad.

First, we show, in Fig. 14.4, the redundancy-resolution results with joint physical limits considered (i.e., $\lambda = 0.9$ and $\kappa_\alpha = \kappa_\beta = 20$) but without considering cyclic motion criterion (i.e., $\gamma = 0$). As seen from the upper left graph of Fig. 14.4, the end-effector of the PUMA560 robot manipulator moves along a "⊟" path, which indicates that the robot manipulator completes the desired path-tracking task. From the upper left graph of Fig. 14.4, we can also see that such a solution is not repetitive, because the final and initial states of the manipulator are not equal. Thus, if such a non-repetitive solution is exploited to control the PUMA560 robot manipulator, then an additional self-motion readjustment is needed, which would be of low efficiency in engineering applications.

Second, for comparison as well as for illustration, the inverse kinematics problem is finally solved via the proposed acceleration-level RMP scheme (14.7)–(14.11) with both joint physical limits and RMP criterion considered (i.e., $\lambda = 0.9$, $\kappa_\alpha = \kappa_\beta = 20$ and $\gamma = 5$). The corresponding simulation results are shown in the rest graphs of Fig. 14.4. As seen from the upper right graph of Fig. 14.4, the end-effector of

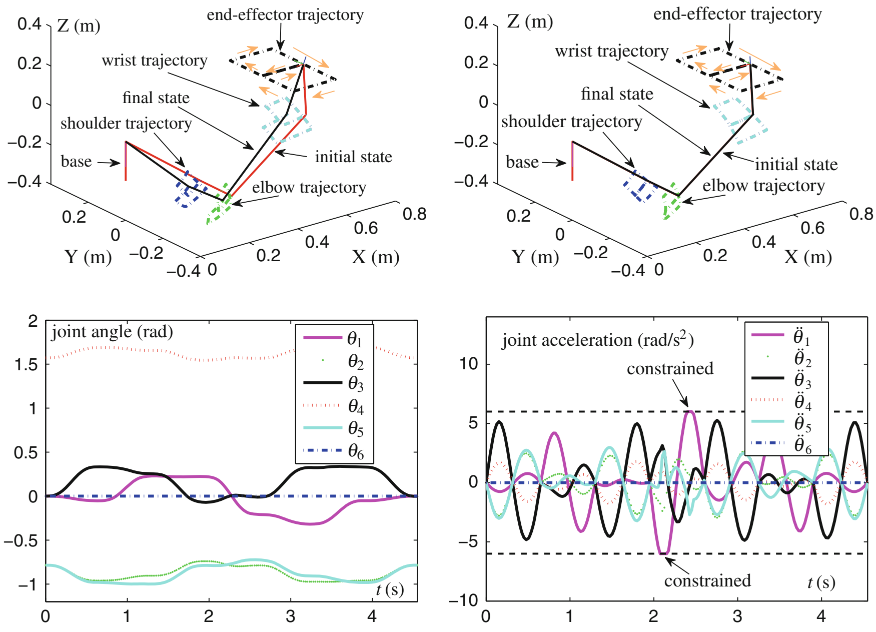


Fig. 14.4 PUMA560 end-effector tracks a path of an East-Asian character looking like "⊟" synthesized by the scheme without and with RMP criterion considered

PUMA560 robot manipulator tracks a "⊟" path, which is sufficiently close to the desired one. In addition, this solution is repetitive and applicable for PUMA560 robot manipulator working with a cyclic motion requirement, because the final and initial states of the robot manipulator coincide with each other as shown in the upper right graph of Fig. 14.4, which can also be seen from the θ profiles in the lower left graph of Fig. 14.4. Furthermore, as seen from the lower two graphs of Fig. 14.4 and Table 14.1, the joint angles and joint accelerations are kept within their limited ranges. Note that the joint velocities are also within their ranges and the corresponding figure is omitted due to the space limitation. It is worth pointing out that the maximum component of the robot end-effector's positioning error is also small during the "⊟" path tracking task execution, i.e., less than 8×10^{-4} m. The efficacy of the proposed RMP scheme (14.7)–(14.11) at the joint-acceleration level subject to joint-angle limits, joint-velocity limits and joint-acceleration limits is thus well substantiated.

14.4.3 "V" Path Path Tracking

In order to further demonstrate the efficacy and the general applicability of the proposed acceleration-level RMP scheme (14.7)–(14.11) as well as the corresponding

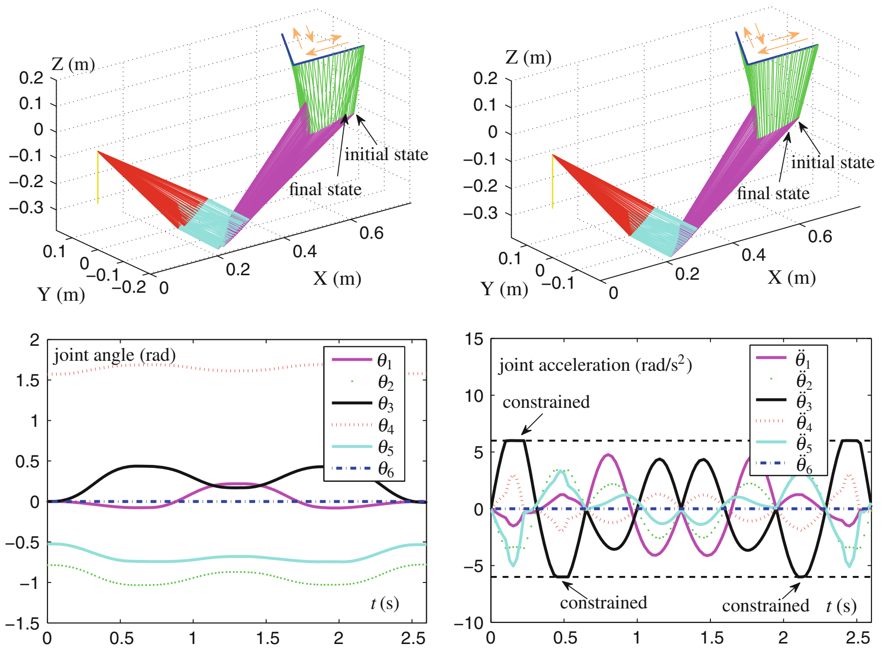


Fig. 14.5 PUMA560 end-effector tracks the path of the initial letter "V" of English word "VICTORY" synthesized by scheme (14.7)–(14.11) without and with RMP criterion considered (i.e., corresponding to $\gamma = 0$ and $\gamma \neq 0$)

QP solver [i.e., PDNN (14.18)], the initial letter "V" writing of English word "VICTORY" is also performed based on PUMA560 robot manipulator in this subsection.

Figure 14.5 illustrates the simulation results of PUMA560 robot manipulator's end-effector tracking the "V" path. Specifically, the upper left graph of Fig. 14.5 shows the simulation result with joint physical limits considered but without considering RMP criterion (i.e., $\gamma = 0$). From such a figure, we can see that the initial and final states of the robot manipulator do not match. For comparison, the simulation results with both joint physical limits and RMP criterion considered are shown in the rest graphs of Fig. 14.5. These simulation results illustrate that, by applying the proposed acceleration-level RMP scheme (14.7)–(14.11) to the PUMA560 robot manipulator, the joint drift phenomenon is remedied, all joint variables [i.e., joint angle $\theta(t)$, joint velocity $\dot{\theta}(t)$ and joint acceleration $\ddot{\theta}(t)$] are kept within their limited ranges, and that the positioning error of the robot end-effector is small. These comparisons of simulation results further substantiate the efficacy and accuracy of the proposed RMP scheme (14.7)–(14.11), as well as the effectiveness of the PDNN (14.18) as a QP solver.

14.4.4 Comparisons with Velocity-Level RMP Scheme

To substantiate the superiority of the proposed acceleration-level RMP scheme (14.7)–(14.11), comparisons between the RMP schemes at the joint-acceleration and joint-velocity levels are shown in this subsection. As presented in [6, 17, 18], the velocity-level RMP scheme is formulated as follows:

$$\text{minimize} \quad (\dot{\theta} + \rho)^{\mathrm{T}}(\dot{\theta} + \rho)/2 \tag{14.19}$$

$$\text{subject to} \quad J(\theta)\dot{\theta} = \dot{\mathbf{r}}_{\mathrm{d}}, \tag{14.20}$$

$$\theta^{-} \leqslant \theta \leqslant \theta^{+}, \tag{14.21}$$

$$\dot{\theta}^{-} \leqslant \dot{\theta} \leqslant \dot{\theta}^{+}, \tag{14.22}$$

where $\rho = \gamma(\theta(t) - \theta(0))$, and $\dot{\mathbf{r}}_{\mathrm{d}} \in \mathbb{R}^m$ is the time derivative of the desired end-effector path $\mathbf{r}_{\mathrm{d}}$.

Comparing these two different level RMP schemes, we find that the proposed acceleration-level RMP scheme (14.7)–(14.11) can incorporate the joint-acceleration limits into the scheme formulation readily, but the presented velocity-level RMP scheme (14.19)–(14.22) can not do it. In other words, the redundancy-resolution results generated by the velocity-level scheme probably exceed the physical acceleration limits sometimes. To explain this point, without loss of generality, we still take tracking the pentagram-path for example. The initial state $\theta(0)$, task duration T_{d} and the corresponding design parameters κ_α, κ_β, λ, and γ are set the same as before. In this comparison example, $l = 0.12$ m. The acceleration profiles of the RMP scheme resolved, respectively, at the velocity level and acceleration level when the end-effector of PUMA560 robot manipulator tracks the pentagram path

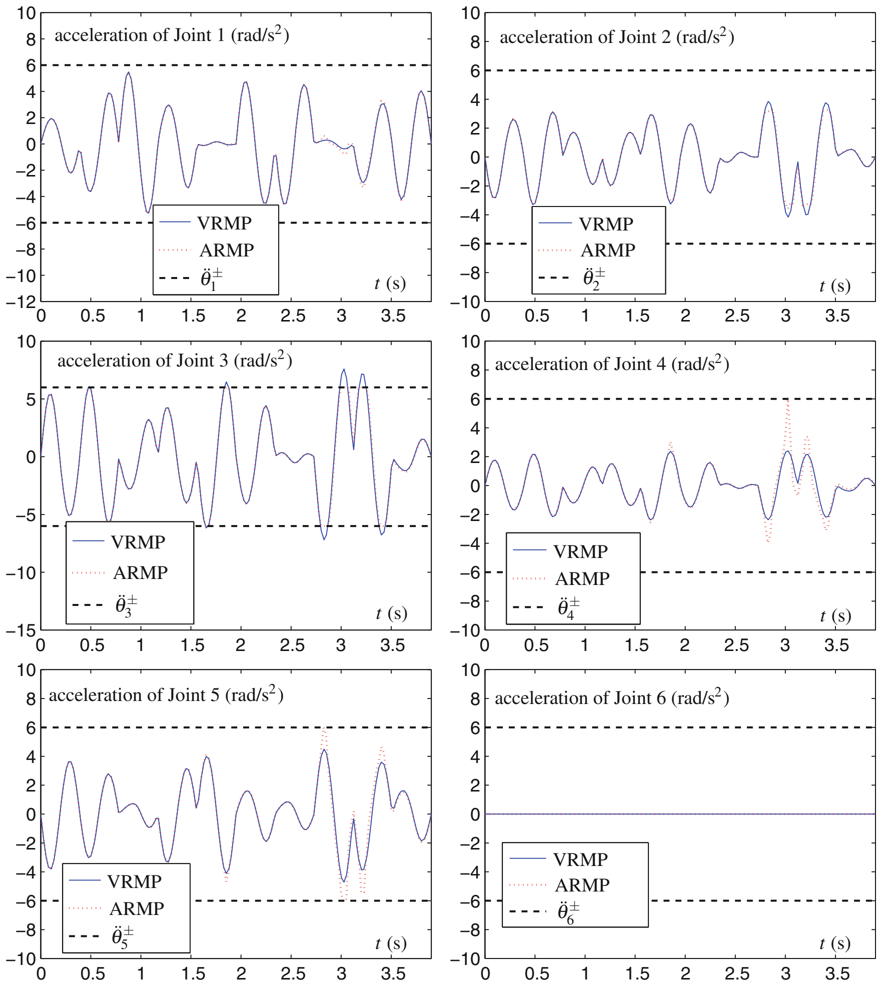


Fig. 14.6 Profiles of the joint acceleration $\ddot{\theta} \in \mathbb{R}^6$: comparisons between the RMP schemes, respectively, resolved at the joint-velocity and joint-acceleration levels

are presented in Fig. 14.6. Note that, in Fig. 14.6, VRMP denotes the RMP scheme at the joint-velocity level [i.e., (14.19)–(14.22)] and ARMP denotes the RMP scheme at the joint-acceleration level [i.e., (14.7)–(14.11)]. As shown in Fig. 14.6, all the joint-acceleration profiles generated by ARMP, denoted by the red dot line, are constrained within the acceleration limits $\ddot{\theta}_i^{\pm}$ (with $i = 1, 2, \ldots, 6$) denoted by the black dash line. Even some joint accelerations (i.e., $\ddot{\theta}_3$, $\ddot{\theta}_4$ and $\ddot{\theta}_5$) reach their limits in some durations, the bound constraints keep them within their physical limits. For instance, $\ddot{\theta}_3$ reaches its upper or lower limit many times (e.g., during [1.828, 1.875], [2.801, 2.875], [2.978, 3.075], [3.175, 3.253] and [3.378, 3.428] s) as shown in Fig. 14.6, and it stops increasing in these durations. This illustrates that the bound constraint (14.11) is activated effectively. On the contrary, joint acceleration $\ddot{\theta}_3$ indirectly resolved by VRMP, denoted by the blue solid line, exceeds the limits $\ddot{\theta}_3^{\pm}$ because the joint-acceleration limits have not been considered in the VRMP. In actual application, the case of exceeding physical limits is unallowed, because it may lead to the acceleration saturation and even destroy the physical robot. From this point, the RMP scheme at the joint-acceleration level is readily applied and preferred. Another advantage of the proposed acceleration-level RMP scheme (14.7)–(14.11) is that, in applications, some robot manipulators are controlled by acceleration (such as the robot in [24]). This kind of robot cannot directly use the redundancy-resolution results generated by the presented velocity-level RMP scheme (14.19)–(14.22), and it is less efficient to transform the resolution results (i.e., joint velocity $\dot{\theta}$) of the velocity-level RMP scheme to joint acceleration $\ddot{\theta}$.

In summary, compared with the presented velocity-level RMP scheme (14.19)–(14.22), the proposed acceleration-level RMP scheme (14.7)–(14.11) is safer and more applicable (showing again the successful application of the presented ZD approach to RMP of fixed-base redundant robot manipulators).

14.5 Summary

In this chapter, by defining two different ZFs and by exploiting the ZD design formula, the acceleration-level RMP performance index (14.9) has been proposed, developed, and investigated. Based on such a performance index, the acceleration-level RMP scheme (14.7)–(14.11) has been further presented and investigated to remedy the joint-angle drift phenomenon of fixed-base redundant robot manipulators, which is reformulated as a QP (14.15)–(14.17) and then is solved by PDNN (14.18). Computer simulation results based on PUMA560 robot manipulator performing three different types of end-effector path-tracking tasks have substantiated well the effectiveness, accuracy, and safety of the proposed acceleration-level RMP scheme for physically-constrained redundant robot manipulators, and more importantly, have shown the application prospect of the presented ZD approach to robotic redundancy resolution.

References

1. Janabi-Sharifi F, Hassanzadeh I (2011) Experimental analysis of mobile-robot teleoperation via shared impedance control. IEEE Trans Syst Man Cybern B 41(2):591–606
2. Su J, Xie W (2011) Motion planning and coordination for robot systems based on representation space. IEEE Trans Syst Man Cybern B 41(1):248–299
3. Guo D, Zhang Y (2014) Li-function activated ZNN with finite-time convergence applied to redundant-manipulator kinematic control via time-varying Jacobian matrix pseudoinversion. Appl Soft Comput 24:158–168

4. Sun C, Xu WL, Bronlund JE, Morgenstern M (2014) Dynamics and compliance control of a linkage robot for food chewing. IEEE Trans Ind Electron 61(1):377–386
5. Guo D, Zhang Y (2014) Acceleration-level inequality-based MAN scheme for obstacle avoidance of redundant robot manipulators. IEEE Trans Ind Electron 61(12):6903–6914
6. Zhang Y, Zhang Z (2013) Repetitive motion planning and control of redundant robot manipulators. Springer, New York
7. Okadome Y, Nakamura Y, Ishiguro H (2014) Predictive control method for a redundant robot using a non-parametric predictor. Adv Robot 28(10):647–657
8. Zhang Y, Wang J, Xia Y (2003) A dual neural network for redundancy resolution of kinematically redundant manipulators subject to joint limits and joint velocity limits. IEEE Trans Neural Netw 14(3):658–667
9. Zhang Z, Zhang Y (2012) Acceleration-level cyclic-motion generation of constrained redundant robots tracking different paths. IEEE Trans Syst Man Cybern B 42(4):1257–1269
10. Guo D, Zhang Y (2012) A new inequality-based obstacle-avoidance MVN scheme and its application to redundant robot manipulators. IEEE Trans Syst Man Cybern C 42(6):1326–1340
11. Maciekewski AA, Klein CA (1985) Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments. Int J Robot Res 4(3):109–117
12. Cheng FT, Sheu RJ, Chen TH (1995) The improved compact QP method for resolving manipulator redundancy. IEEE Trans Syst Man Cybern 25(11):1521–1530
13. Zhang Z, Zhang Y (2013) Variable joint-velocity limits of redundant robot manipulators handled by quadratic programming. IEEE/ASME Trans Mech 18(2):674–686
14. Tchon K, Jakubiak J (2005) A repeatable inverse kinematics algorithm with linear invariant subspaces for mobile manipulators. IEEE Trans Syst Man Cybern B 35(5):1051–1057
15. Siciliano B, Khatib O (2008) Springer handbook of robotics. Springer, Heidelberg
16. Siciliano B, Sciavicco L, Villani L, Oriolo G (2009) Robotics: modelling, planning and control. Springer, London
17. Zhang Y, Tan Z, Chen K, Yang Z, Lv X (2009) Repetitive motion of redundant robots planned by three kinds of recurrent neural networks and illustrated with a four-link planar manipulator's straight-line example. Robot Auton Syst 57(6–7):645–651
18. Zhang Y, Lv X, Li Z, Yang Z, Chen K (2008) Repetitive motion planning of PA10 robot arm subject to joint physical limits and using LVI-based primal-dual neural network. Mechatronics 18(9):475–485
19. Cai B, Zhang Y (2010) Bi-criteria optimal control of redundant robot manipulators using LVI-based primal-dual neural network. Optim Control Appl Methods 31(3):213–229
20. Roberts RG, Maciejewski AA (1993) Repeatable generalized inverse control strategies for kinematically redundant manipulators. IEEE Trans Autom Control 38(5):689–698
21. Shamir T, Yomdin Y (1988) Repeatability of redundant manipulators: mathematical solution of the problem. IEEE Trans Autom Control 33(11):1004–1009
22. Ge SS, Zhang Y, Lee TH (2004) An acceleration-based weighting scheme for minimum-effort inverse kinematics of redundant manipulators. In: Proceedings of the IEEE international symposium on intelligent control, pp 275–280
23. Zhang Y, Yi C (2011) Zhang neural networks and neural-dynamic method. Nova Science Publishers, New York
24. Xue F, Hou Z, Deng H (2011) Balance control for an acrobat. In: Proceedings of the IEEE international conference on control decision, pp 3426–3429

Chapter 15
Application to Mobile Robot RMP

Abstract In this chapter, the application of the ZD approach is further investigated to the velocity-level RMP of mobile redundant robot manipulators. That is, by introducing three different ZFs and by exploiting the ZD design formula, we propose, develop, and investigate a velocity-level RMP performance index. Then, based on such a performance and with physical limits considered, the resultant RMP scheme is presented and investigated to remedy the joint-angle drift phenomenon of mobile redundant robot manipulators. Such a scheme is reformulated as a QP, which is solved by a numerical algorithm. With two path-tracking examples, simulation results based on a wheeled mobile robot manipulator substantiate well the effectiveness and accuracy of the proposed velocity-level RMP scheme (as well as show the application prospect of the presented ZD approach once again).

15.1 Introduction

As presented in Chap. 14, fixed-base redundant robot manipulators (e.g., the PUMA 560 robot manipulator) have long been studied and widely applied in factory automation [1–8]. In addition, many techniques have been developed and investigated for motion planning of fixed-base redundant robot manipulators. The most popular method is to apply the pseudoinverse formulation for obtaining a general solution at the joint-velocity and/or joint-acceleration level, which contains a minimum-norm particular solution and a homogeneous solution [9–12]. However, this method has the generally undesirable property that repetitive end-effector motions do not necessarily yield repetitive joint motions. Thus, the manipulator's behavior is difficult to predict when the end-effector traces a closed path in its workspace. In addition, it is less efficient to readjust the manipulator's configuration after every cycle via self-motion such that the joint angles return to their initial values. In the last three decades, a large number of research on the topic of repetitive motion have been produced [5, 13–18] (see also Chap. 14).

With the evolution of the complex technological society and the introduction of new notions and innovative theoretical tools in the field of intelligent systems, mobile manipulators are attracting significant interest in the industrial, military, and public

Y. Zhang and D. Guo, *Zhang Functions and Various Models*,
DOI 10.1007/978-3-662-47334-4_15

service communities [19–22], because of their large-scale mobility and manipulation abilities, as compared to these of the fixed-base manipulators. In general, a mobile manipulator is a robotic device composed of a mobile platform and a stationary robot manipulator fixed to the platform. However, the integration of a redundant robot manipulator and a mobile platform gives rise to many new difficulties; for example, how to coordinate a given task into fine motions to be carried out by the robot manipulator and the gross motions to be achieved by the mobile platform; and how to define the repeatability of a redundancy-resolution scheme for a mobile robot manipulator. It is well known that, for a stationary robot manipulator, an redundancy-resolution scheme is called repetitive, if it maps closed paths in the task space to closed trajectories in the configuration space (see also Chap. 14). However, for a mobile robot manipulator, if the mobile platform does not return to the initial position, a repetitive redundancy-resolution scheme for a stationary robot manipulator is no longer fit for a mobile robot manipulator. Note that repetitive motion control of mobile robot manipulators starts to play a more and more important role in practical applications, which urgently requires an effective scheme for solving the non-repetitive problem of mobile robot manipulators. In [23], Tchoń was the first to introduce the concept of repeatability of inverse kinematics algorithms for mobile robot manipulators by exploiting the endogenous configuration space approach; and further presented repeatable inverse kinematics algorithms [24–26], which provides an insight into the mechanism of repeatability. However, among these inverse kinematics algorithms for repeatability of mobile robot manipulators, the physical constraints (e.g., joint-angle limits and joint-velocity limits) are usually not taken into account. If these physical limits are not considered, a saturation may occur in some cases. Thus, these schemes may be less effective to control mobile robot manipulators for generating cyclic motion.

In this chapter, based on the presented ZD approach, we propose and investigate a novel redundancy-resolution scheme at the velocity level to achieve the RMP purpose of mobile redundant robot manipulators (with physical constraints considered). Specifically, by introducing three different ZFs and by exploiting the ZD design formula [27], a velocity-level RMP performance index is proposed, developed and investigated. To the best of the authors' knowledge, such a new RMP performance index for mobile redundant robot manipulators has never been investigated before by others. Then, a novel RMP scheme is presented by combining the proposed performance index, physical constraints, and integrated kinematical equations of mobile robot manipulators, and further reformulated as a QP subject to equality and bound constraints. As an illustrative example, a wheeled mobile robot manipulator is studied, which is composed of a mobile platform driven by two independent wheels, and a six-DOF spatial robot manipulator mounted on the platform. Tracking-path tasks based on such a wheeled mobile robot manipulator are performed, which further substantiate the efficacy of the proposed RMP scheme and show once again the application prospect of the presented ZD approach. Besides, it is worth pointing out here that, in our previous book [5], the investigation of velocity-level RMP is presented only for fixed-base redundant robot manipulators. By contrast, in this book (or specifically, in this chapter), we focus on investigating the velocity-level RMP for mobile redundant robot manipulators. Thus, the velocity-level RMP investigation in this chapter can be viewed as an extension of that in [5] (i.e., from the fixed-base robot to the mobile one).

15.2 RMP Performance Index Derived via Different ZFs

In this section, the presented ZD approach is applied to deriving the velocity-level RMP performance index for mobile redundant robot manipulators.

15.2.1 Kinematics Modeling of Mobile Robot Manipulators

In this subsection, a wheeled mobile redundant robot manipulator [21, 22, 28, 29] is developed to lay a basis for further discussion and to substantiate the efficacy of the proposed velocity-level RMP scheme (see Sect. 15.4).

The computer-aided design model of the mobile redundant robot manipulator is shown in Fig. 15.1, which is composed of a wheeled mobile platform and a six-DOF spatial robot manipulator. The mobile platform includes two independent drive wheels and two omnidirectional passive supporting wheels. In this chapter, only the end-effector position is considered, and thus the robot manipulator becomes a functionally redundant robot manipulator. As for such a mobile redundant robot manipulator, the integrated kinematics at the velocity level is obtained in the following form (with the detailed derivation being shown in [22]):

$$\dot{\mathbf{r}}_{\mathrm{w}} = J(\vartheta)\dot{\mathbf{q}}, \tag{15.1}$$

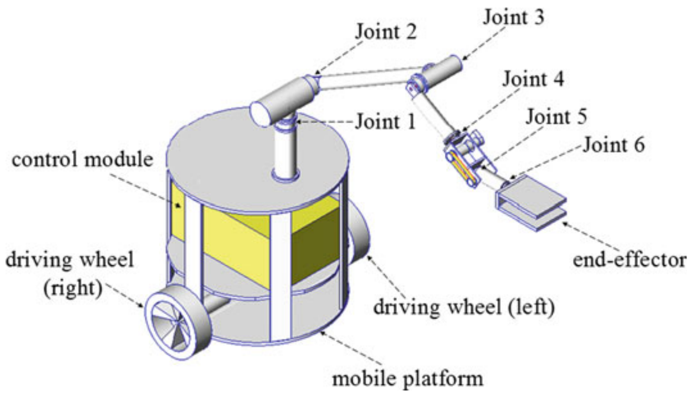


Fig. 15.1 The computer-aided design model of the mobile redundant robot manipulator

where $\dot{\mathbf{r}}_\mathrm{w} \in \mathbb{R}^m$ is the time derivative of the manipulator's end-effector position-and-orientation vector $\mathbf{r}_\mathrm{w}$ in Cartesian space with respect to the world coordinate frame. In addition, $J(\vartheta) \in \mathbb{R}^{m\times(2+n)}$ is the Jacobian matrix with vector $\vartheta = [\phi, \theta^\mathrm{T}]^\mathrm{T} \in \mathbb{R}^{1+n}$. Besides, $\phi \in \mathbb{R}$ is the heading angle of the mobile platform, and $\theta \in \mathbb{R}^n$ denotes the joint-angle vector of the six-DOF robot manipulator. Note that we define $\varphi = [\varphi_\mathrm{l}, \varphi_\mathrm{r}]^\mathrm{T} \in \mathbb{R}^2$ as the driving wheel angle (angular position) vector, with φ_l and φ_r being the angles of left and right driving wheel, respectively. Thus, we have the combined angle vector $\mathbf{q} = [\varphi^\mathrm{T}, \theta^\mathrm{T}]^\mathrm{T} \in \mathbb{R}^{2+n}$ for the presented mobile redundant robot manipulator, and its time derivative $\dot{\mathbf{q}} = [\dot{\varphi}^\mathrm{T}, \dot{\theta}^\mathrm{T}]^\mathrm{T}$ (i.e., the combined velocity vector).

15.2.2 Velocity-Level RMP Performance Index

In order to achieve the RMP purpose of the mobile robot manipulators, let us consider three factors of mobile robot manipulators, i.e., the joint angle of the robot manipulator $\theta \in \mathbb{R}^n$, the rotational angle of the mobile platform $\phi \in \mathbb{R}$ and the location of the robot manipulator on the mobile platform $\mathbf{p}_\mathrm{C} = [x_\mathrm{C}, y_\mathrm{C}]^\mathrm{T} \in \mathbb{R}^2$. Evidently, the mobile robot manipulator can return to the original state if and only if these three variables return to their initial positions, when the end-effector of mobile robot manipulators performs a closed trajectory. That is to say, the repetitive motion of mobile robot manipulators is equivalent to the repetitive motion of variables θ, ϕ, and $\mathbf{p}_\mathrm{C}$. Thus, by following the presented ZD approach, the repetitive motion of θ, ϕ, and $\mathbf{p}_\mathrm{C}$ can be achieved through the following design steps.

- First, to achieve the RMP purpose of the mobile robot manipulators, we define three different ZFs as follows:

$$\mathbf{e}(t) = \theta - \theta(0) \in \mathbb{R}^n, \tag{15.2}$$

$$e(t) = \sin\phi - \sin\phi(0) \in \mathbb{R}, \tag{15.3}$$

$$\mathbf{e}(t) = \mathbf{p}_\mathrm{C} - \mathbf{p}_\mathrm{C}(0) \in \mathbb{R}^2, \tag{15.4}$$

  where $\theta(0)$, $\phi(0)$, and $p_\mathrm{C}(0)$ denote the initial states of θ, ϕ and p_C, respectively. It is worth pointing out that, when the mobile platform moves circularly (i.e., the heading angle ϕ makes 360° turns), the heading angle can return to the initial position. In this situation, the value of the heading angle is $\phi = \phi(0) + 2k\pi$ ($k = \pm1, \pm2, \ldots$). If $\phi - \phi(0)$ is used [instead of $\sin\phi - \sin\phi(0)$], the restrictions become harsh and the resultant RMP scheme is difficult to realize. Thus, the sine function is exploited, and the simulation and modeling of the velocity-level RMP scheme become easy.
- Second, by combining the ZD design formula [27] and the above three ZFs, respectively [i.e., (4.2) corresponding to (15.2), (15.4), and (1.2) corresponding to (15.3)], three resultant differential equations are obtained as follows:

$$\dot{\theta} + \gamma_1(\theta - \theta(0)) = \mathbf{0} \in \mathbb{R}^n, \tag{15.5}$$

$$\dot{\phi}\cos\phi + \gamma_2(\sin\phi - \sin\phi(0)) = 0 \in \mathbb{R}, \tag{15.6}$$

$$\dot{\mathbf{p}}_{\mathrm{C}} + \gamma_3(\mathbf{p}_{\mathrm{C}} - \mathbf{p}_{\mathrm{C}}(0)) = \mathbf{0} \in \mathbb{R}^2. \tag{15.7}$$

For the above equations, $\dot{\theta} \in \mathbb{R}^n$ is the joint-velocity vector of the robot manipulator, $\dot{\phi} \in \mathbb{R}$ is the heading velocity of the mobile platform (being the time derivative of ϕ), and $\dot{\mathbf{p}}_{\mathrm{C}} \in \mathbb{R}^2$ is the time derivative of $\mathbf{p}_{\mathrm{C}}$. In addition, design parameters $\gamma_1 > 0 \in \mathbb{R}$, $\gamma_2 > 0 \in \mathbb{R}$, and $\gamma_3 > 0 \in \mathbb{R}$ are used for achieving the RMP purpose. In addition, one can prove theoretically that, in equations (15.5)–(15.7), as $t \to \infty$, θ, $\sin\phi$ and $\mathbf{p}_{\mathrm{C}}$ can converge to their initial states globally and exponentially.

- Third, in (15.1), the variables of the velocity-level integrated kinematics of the mobile root manipulator only include $\dot{\varphi}$ and $\dot{\theta}$ (i.e., $\dot{\mathbf{q}} = [\dot{\varphi}^{\mathrm{T}}, \dot{\theta}^{\mathrm{T}}]^{\mathrm{T}}$), so Eqs. (15.5)–(15.7) have to be converted into a $\dot{\mathbf{q}}$-based matrix-vector equation for meeting the needs of the simulation modeling. Note that, as for the mobile platform shown in Fig. 15.1, we have
$$A\dot{\varphi} = \dot{\phi} \quad \text{and} \quad B\dot{\varphi} = \dot{\mathbf{p}}_{\mathrm{C}},$$
where matrices A and B are defined respectively as follows:
$$A = \frac{r}{2b}\begin{bmatrix} -1 \\ 1 \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^{1\times 2}, \text{ and } B = \frac{r}{2}\begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix}\begin{bmatrix} 1 & 1 \\ -d/b & d/b \end{bmatrix} \in \mathbb{R}^{2\times 2},$$
with r being the radius of the drive wheels, b being the distance between the drive wheels and the middle point of the two-drive wheel axis (denoted as P_0), and d being the distance between the point connecting the robot manipulator and the mobile platform (corresponding to $\mathbf{p}_{\mathrm{C}}$) and point P_0. Thus, based on the above analysis, Eqs. (15.6) and (15.7) are merged into an equation in the form of
$$\begin{bmatrix} B \\ A\cos\phi \end{bmatrix}\dot{\varphi} + \begin{bmatrix} \gamma_3(\mathbf{p}_{\mathrm{C}} - \mathbf{p}_{\mathrm{C}}(0)) \\ \gamma_2(\sin\phi - \sin\phi(0)) \end{bmatrix} = \mathbf{0} \in \mathbb{R}^3. \tag{15.8}$$
- Fourth, by defining $\mathbf{a} = [\gamma_3(\mathbf{p}_{\mathrm{C}} - \mathbf{p}_{\mathrm{C}}(0)), \gamma_2(\sin\phi - \sin\phi(0))]^{\mathrm{T}} \in \mathbb{R}^3$, $C = [B^{\mathrm{T}}, A^{\mathrm{T}}\cos\phi]^{\mathrm{T}} \in \mathbb{R}^{3\times 2}$, and $\mathbf{c} = \gamma_1[\theta - \theta(0)] \in \mathbb{R}^n$, Eqs. (15.5) and (15.8) are further combined into a unified matrix-vector equation as follows:
$$D\dot{\mathbf{q}} + \mathbf{z} = \mathbf{0} \in \mathbb{R}^{n+3}, \tag{15.9}$$
where matrix D and vector z are defined respectively as
$$D = \begin{bmatrix} C & 0 \\ 0 & I \end{bmatrix} \in \mathbb{R}^{(n+3)\times(n+2)} \text{ and } \mathbf{z} = \begin{bmatrix} \mathbf{a} \\ \mathbf{c} \end{bmatrix} \in \mathbb{R}^{n+3},$$

with $I \in \mathbb{R}^{n \times n}$ being the identity matrix. Therefore, from the above derivation, one knows that solving (15.5)–(15.7) is equivalent to solving (15.9). In addition, when (15.9) is solved, the resultant solutions can achieve the RMP purpose of mobile redundant robot manipulators.

- Finally, because the end-effector motion-trajectory requirement has to be considered and physical constraints always exist in mobile manipulators, it is better to minimize $\|D\dot{\mathbf{q}} + \mathbf{z}\|_2^2/2$, rather than use $D\dot{\mathbf{q}} + \mathbf{z} = \mathbf{0}$ directly. Thus, we obtain

$$\|D\dot{\mathbf{q}} + \mathbf{z}\|_2^2/2 = (D\dot{\mathbf{q}} + \mathbf{z})^{\mathrm{T}}(D\dot{\mathbf{q}} + \mathbf{z})/2, \tag{15.10}$$

which is the velocity-level RMP performance index for mobile redundant robot manipulators.

In summary, by using the ZD approach, we have developed the RMP performance index (15.10) at the velocity level (showing the application prospect of such a ZD approach once again). Note that, by minimizing the velocity-level performance index (15.10) [i.e., “minimize $(D\dot{\mathbf{q}} + \mathbf{z})^{\mathrm{T}}(D\dot{\mathbf{q}} + \mathbf{z})/2$”], the RMP purpose is thus achieved for mobile redundant robot manipulators.

15.3 Scheme and QP Formulations

In this section, based on the proposed performance index (15.10), a novel velocity-level RMP scheme is further developed and investigated for mobile redundant robot manipulators. In addition, such an acceleration-level RMP scheme is reformulated as a QP, which is solved by a numerical algorithm.

15.3.1 Velocity-Level RMP Scheme

For mobile redundant robot manipulators, with physical limits considered, the velocity-level RMP scheme is proposed as follows:

$$\text{minimize} \quad (D\dot{\mathbf{q}} + \mathbf{z})^{\mathrm{T}}(D\dot{\mathbf{q}} + \mathbf{z})/2 \tag{15.11}$$

$$\text{subject to} \quad J(\vartheta)\dot{\mathbf{q}} = \dot{\mathbf{r}}_{\mathrm{dw}}, \tag{15.12}$$

$$\mathbf{q}^- \leqslant \mathbf{q} \leqslant \mathbf{q}^+, \tag{15.13}$$

$$\dot{\mathbf{q}}^- \leqslant \dot{\mathbf{q}} \leqslant \dot{\mathbf{q}}^+, \tag{15.14}$$

where $\dot{\mathbf{r}}_{\mathrm{dw}} \in \mathbb{R}^m$ denotes the time derivative of the desired end-effector path $\mathbf{r}_{\mathrm{dw}}$. In addition, $\mathbf{q}^-$ and $\mathbf{q}^+$ denote respectively the lower and upper limits of the combined angle vector $\mathbf{q}$. Furthermore, $\dot{\mathbf{q}}^-$ and $\dot{\mathbf{q}}^+$ denote respectively the lower and upper limits of the combined velocity vector $\dot{\mathbf{q}}$. Note that the physical limits (i.e., $\mathbf{q}^-$, $\mathbf{q}^+$,

Table 15.1 The physical limits used in the wheeled mobile robot manipulator shown in Fig. 15.1

i	1	2	3	4	5	6	7	8
q_i^- (rad)	$-\infty$	$-\infty$	−0.5236	−0.1745	−6.2832	−6.2832	−6.2832	−6.2832
q_i^+ (rad)	$+\infty$	$+\infty$	+0.5236	+2.3562	+6.2832	+6.2832	+6.2832	+6.2832
$\dot{q}_i^-$ (rad/s)	−30.0000	−30.0000	−3.0000	−3.0000	−3.0000	−3.0000	−3.0000	−3.0000
$\dot{q}_i^+$ (rad/s)	+30.0000	+30.0000	+3.0000	+3.0000	+3.0000	+3.0000	+3.0000	+3.0000

$\dot{\mathbf{q}}^-$ and $\dot{\mathbf{q}}^+$) used in the wheeled mobile robot manipulator shown in Fig. 15.1 are presented in Table 15.1.

15.3.2 QP Reformulation

As the proposed RMP scheme (15.11)–(15.14) is resolved at the combined-vector velocity level (i.e., in terms of $\dot{\mathbf{q}}$), the limited combined angle vector range $[\mathbf{q}^-, \mathbf{q}^+]$ has to be converted into a $\dot{\mathbf{q}}$-based expression. The following conversion technique is adopted:

$$\mu(\mathbf{q}^- - \mathbf{q}) \leqslant \dot{\mathbf{q}} \leqslant \mu(\mathbf{q}^+ - \mathbf{q}),$$

where $\mu > 0 \in \mathbb{R}$ is used to scale the feasible region of $\dot{q}$. Note that large values of μ may cause sudden deceleration for joints and wheels when the mobile robot manipulator approaches their physical limits, and that, in mathematically, $\mu \geqslant 2\max_{1\leqslant i\leqslant n+2}\{\dot{q}_i^+/(q_i^+ - q_i^-), -\dot{q}_i^-/(q_i^+ - q_i^-)\}$. Therefore, the following new combined bound constraints can be used to replace (15.13) and (15.14):

$$\zeta^- \leqslant \dot{\mathbf{q}} \leqslant \zeta^+,$$

where the ith elements of ζ^- and ζ^+ are, respectively, defined as (with $i = 1, 2, \ldots, n+2$) $\zeta_i^- = \max\{\mu(q_i^- - q_i), \dot{q}_i^-\}$ and $\zeta_i^+ = \min\{\mu(q_i^+ - q_i), \dot{q}_i^+\}$.

By summarizing the above analysis and defining the decision variable vector $\mathbf{x} = \dot{\mathbf{q}} \in \mathbb{R}^{n+2}$, the proposed RMP scheme (15.11)–(15.14) for mobile redundant robot manipulators is reformulated as the following QP, in view of $(D\mathbf{x}+\mathbf{z})^{\mathrm{T}}(D\mathbf{x}+\mathbf{z})/2 = \mathbf{x}^{\mathrm{T}}D^{\mathrm{T}}D\mathbf{x}/2 + \mathbf{z}^{\mathrm{T}}D\mathbf{x} + \mathbf{z}^{\mathrm{T}}\mathbf{z}/2$:

$$\begin{aligned}
\text{minimize} \quad & \mathbf{x}^{\mathrm{T}}W\mathbf{x}/2 + \mathbf{h}^{\mathrm{T}}\mathbf{x} && (15.15)\\
\text{subject to} \quad & C\mathbf{x} = \mathbf{d}, && (15.16)\\
& \zeta^- \leqslant \mathbf{x} \leqslant \zeta^+, && (15.17)
\end{aligned}$$

where $W = D^{\mathrm{T}}D \in \mathbb{R}^{(n+2)\times(n+2)}$, $C = J(\vartheta) \in \mathbb{R}^{m\times(n+2)}$, $\mathbf{h} = D^{\mathrm{T}}\mathbf{z} \in \mathbb{R}^{n+2}$, and $d = \dot{\mathbf{r}}_{\mathrm{dw}} \in \mathbb{R}^m$. As for the above QP, the performance criterion (15.15) is for the RMP purpose and results from the simplification of (15.11). The equality constraint (15.16) [i.e., (15.1) or (15.12)] describes the integrated kinematics relationship of mobile robot manipulators at the combined-vector velocity level. The inequality constraint (15.17) is used to equivalently replace constraints (15.13) and (15.14) by exploiting a conversion technique.

15.3.3 QP Solver

Note that the presented QP formulation [i.e., (15.15)–(15.17)] is the same as the one shown in Chap. 14. Thus, the PDNN (14.18) can also be used to solve such a QP problem (15.15)–(15.17). In this subsection, being different from the PDNN, a numerical algorithm is exploited to solve the presented QP problem [as well as the proposed RMP scheme (15.11)–(15.14)].

In order to solve the QP problem (15.15)–(15.17), guided by [5, 6, 22, 30–32], we define the following vector-valued error function:

$$\mathbf{e}(\mathbf{u}) = \mathbf{u} - P_{\Omega}(\mathbf{u} - (M\mathbf{u} + \mathbf{p})) \in \mathbb{R}^{n+m+2}, \tag{15.18}$$

where the formulations of matrix $M \in \mathbb{R}^{(n+m+2)\times(n+m+2)}$, vectors $\mathbf{u} \in \mathbb{R}^{n+m+2}$ and $\mathbf{p} \in \mathbb{R}^{n+m+2}$ are presented the same as those shown in Chap. 14, and $P_{\Omega}(\cdot)$ is the projection operator.

Let $\mathbb{S} = \{\mathbf{u}^*|\mathbf{u}^*\text{is a zero point of (15.18)}\}$. When the initial value of primal–dual decision variable vector $\mathbf{u}^0 \in \mathbb{R}^{n+m+2}$ is given, for iteration index $k = 0, 1, 2, \ldots,$ if $\mathbf{u}^k \notin \mathbb{S}$, we have the following iteration formula for finding a zero point of (15.18) [as well as for solving QP (15.15)–(15.17)]:

$$\mathbf{u}^{k+1} = \mathbf{u}^k - \frac{\|\mathbf{e}(\mathbf{u}^k)\|_2^2}{\|(M^{\mathrm{T}} + I)\mathbf{e}(\mathbf{u}^k)\|_2^2}(M^{\mathrm{T}} + I)\mathbf{e}(\mathbf{u}^k), \tag{15.19}$$

where $I \in \mathbb{R}^{(n+m+2)\times(n+m+2)}$ is the identity matrix.

An important criterion of measuring the performance of numerical algorithms is their computational complexity. As seen from the above iteration formula (15.19), within one iteration, it only contains $2\hat{\alpha}^2 + 3\alpha + 1$ multiplications and $2\hat{\alpha}^2 + 5\hat{\alpha} - 2$ additions, with $\hat{\alpha} = n + m + 2$. Therefore, the discrete-time QP solver (15.19) has a low computational complexity; i.e., $O(\hat{\alpha}^2)$. In addition, we usually choose the solution obtained in the previous time step as the initial value of the new time step, as such, the discrete-time QP solver (15.19) has better real-time performance. Besides, by following [5, 22, 30, 31], an important theorem about the global convergence of the discrete-time QP solver (15.19) is presented.

Theorem 15.1 *The sequence* $\{\mathbf{u}^k\}$ *generated by the discrete-time solver (15.19) for (15.18) as well as for QP (15.15)–(15.17) satisfies*

$$\|\mathbf{u}^{k+1} - \mathbf{u}^*\|_2^2 \leqslant \|\mathbf{u}^k - \mathbf{u}^*\|_2^2 - \|\mathbf{e}(\mathbf{u}^k)\|_2^2/\|M^{\mathrm{T}} + I\|_{\mathrm{F}}^2$$

for all $u^* \in \mathbb{S}$. *In addition, the sequence* $\{\mathbf{u}^k\}$ *globally converges to a solution* $\mathbf{u}^*$, *of which the first* $(n + 2)$ *elements constitute the optimal solution* $\mathbf{x}^*$ *to QP (15.15)–(15.17).*

15.4 Illustrative Examples

In this section, computer simulations are conducted on the wheeled mobile redundant robot manipulator shown in Fig. 15.1 to substantiate the efficacy of the proposed RMP scheme (15.11)–(15.14). Note that the final error tolerance of $\|\mathbf{e}(\mathbf{u}^k)\|_2$ is set to be 1.0×10^{-6} for discrete-time QP solver (15.19). In the first example, the end-effector of the mobile robot manipulator is expected to track a circular path, and in the second example the end-effector is to follow a Lissajous path. Without loss of generality, we set $\mu = 2$, initial state $\theta(0) = [\pi/12, \pi/3, \pi/3, \pi/3, \pi/3, \pi/3]^{\mathrm{T}}$ rad, and $\phi(0) = x_{\mathrm{C}}(0) = y_{\mathrm{C}}(0) = 0$ rad, for the wheeled mobile robot manipulator.

15.4.1 Circular Path Tracking

In this subsection, the end-effector of the wheeled mobile robot manipulator is expected to track a circular path with radius ψ being 0.3 m. The X-axis, Y-axis, and Z-axis functions of the desired circular path are

$$\begin{cases} \dot{r}_{\mathrm{dwX}}(t) = -\frac{2\pi^2\psi}{T}\sin(2\pi\sin^2(\frac{\pi t}{2T}) + \pi/6)\sin(\frac{\pi t}{2T})\cos(\frac{\pi t}{2T}), \\ \dot{r}_{\mathrm{dwY}}(t) = \frac{2\pi^2\psi}{T}\cos(2\pi\sin^2(\frac{\pi t}{2T}) + \pi/6)\sin(\frac{\pi t}{2T})\cos(\frac{\pi t}{2T}), \\ \dot{r}_{\mathrm{dwZ}}(t) = 0, \end{cases}$$

where task duration $T = 5$ s and $t \in [0, T]$.

Non-repetitive motion First, we show the non-repetitive motion of the mobile robot manipulator. Since γ_1, γ_2 and γ_3 are greater than zero, a non-repetitive motion scheme is produced when $\gamma_1 = \gamma_2 = \gamma_3 = 0$. In this situation, the circular-path tracking results are shown in Fig. 15.2. The upper graph of Fig. 15.2 shows the whole tracking process of the mobile robot manipulator. It follows from such a graph that the final state of the mobile robot manipulator does not return to the initial one, i.e., the solution in this situation is not repetitive. For a better understanding, the middle graph of Fig. 15.2 shows the top view of motion trajectories of the mobile robot manipulator, and the lower graph of Fig. 15.2 shows the motion trajectories of the mobile platform. It follows from such two graphs that the mobile platform is not repetitive after the end-effector completing the circular-path tracking task. Besides, Fig. 15.3 shows profiles of point of junction $[x_{\mathrm{C}}, y_{\mathrm{C}}]^{\mathrm{T}}$, heading angle ϕ, joint angle θ, left and right wheels synthesized by the non-repetitive motion scheme when the mobile robot manipulator tracks the given circular path. From Fig. 15.3, we can see that these variables do not return to their initial values. In engineering applications, this situation may induce a problem wherein the behavior of the mobile robot manipulator is hard to predict. Readjusting the configuration of the manipulator through self-motion is inefficient.

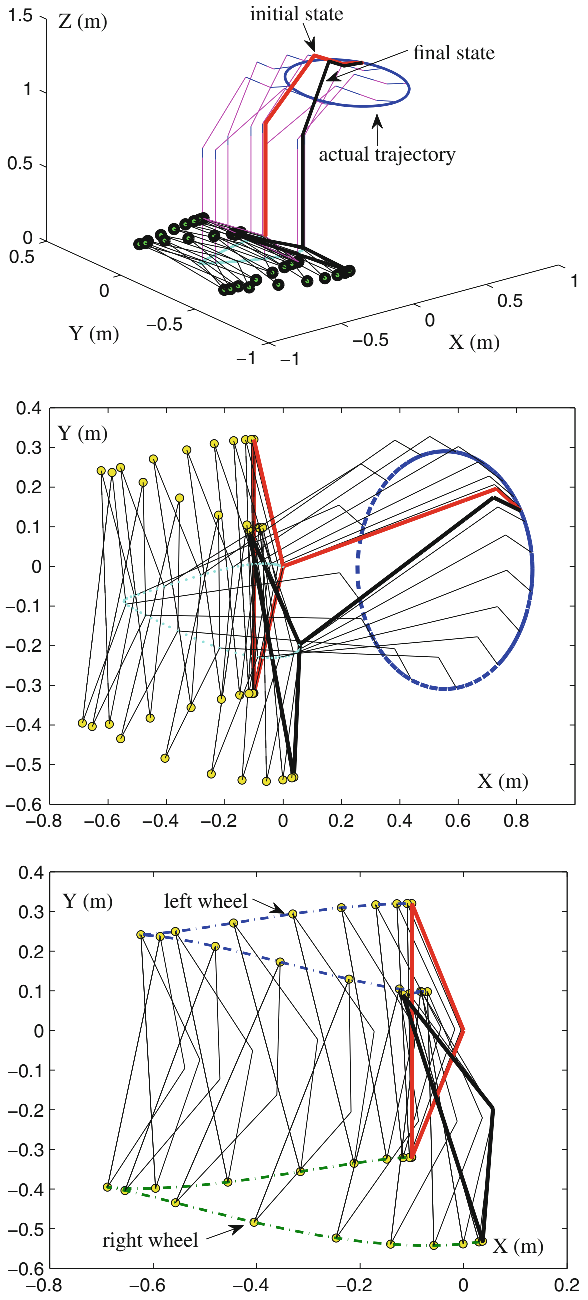


Fig. 15.2 Simulation results when the mobile robot manipulator tracks the given circular path synthesized by the non-repetitive motion scheme

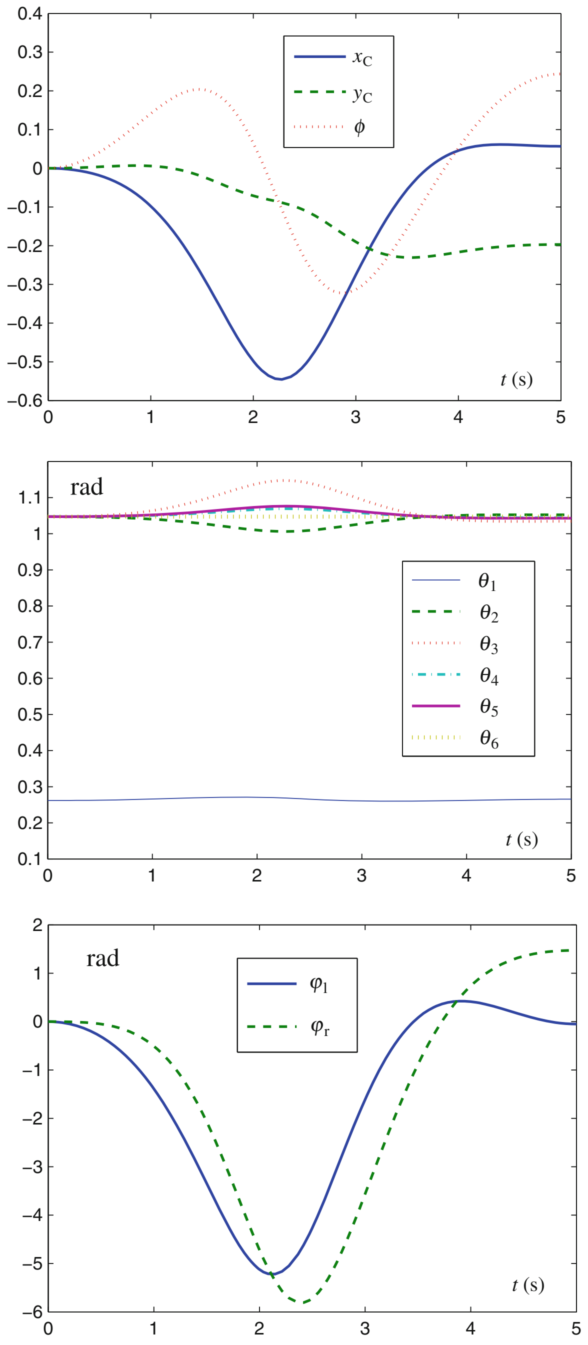


Fig. 15.3 Profiles of point of junction $[x_C, y_C]^T$, heading angle ϕ, joint angle θ, *left* and *right* wheels for the mobile robot manipulator synthesized by the non-repetitive motion scheme

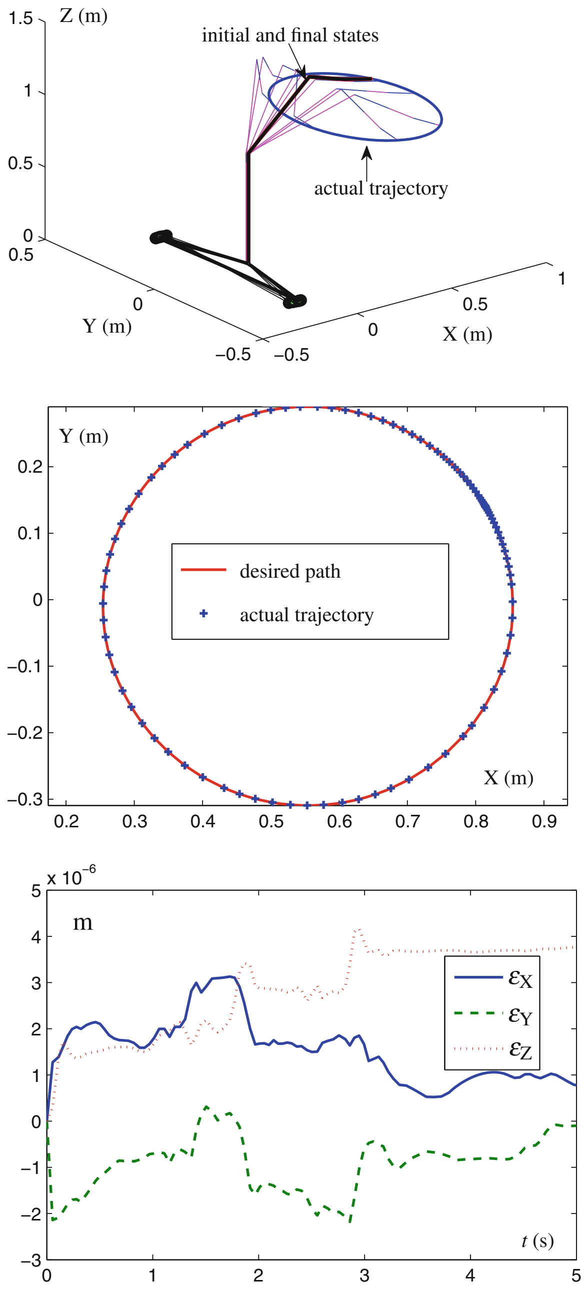


Fig. 15.4 Simulation results when the mobile robot manipulator tracks the given circular path synthesized by the proposed RMP scheme (15.11)–(15.14)

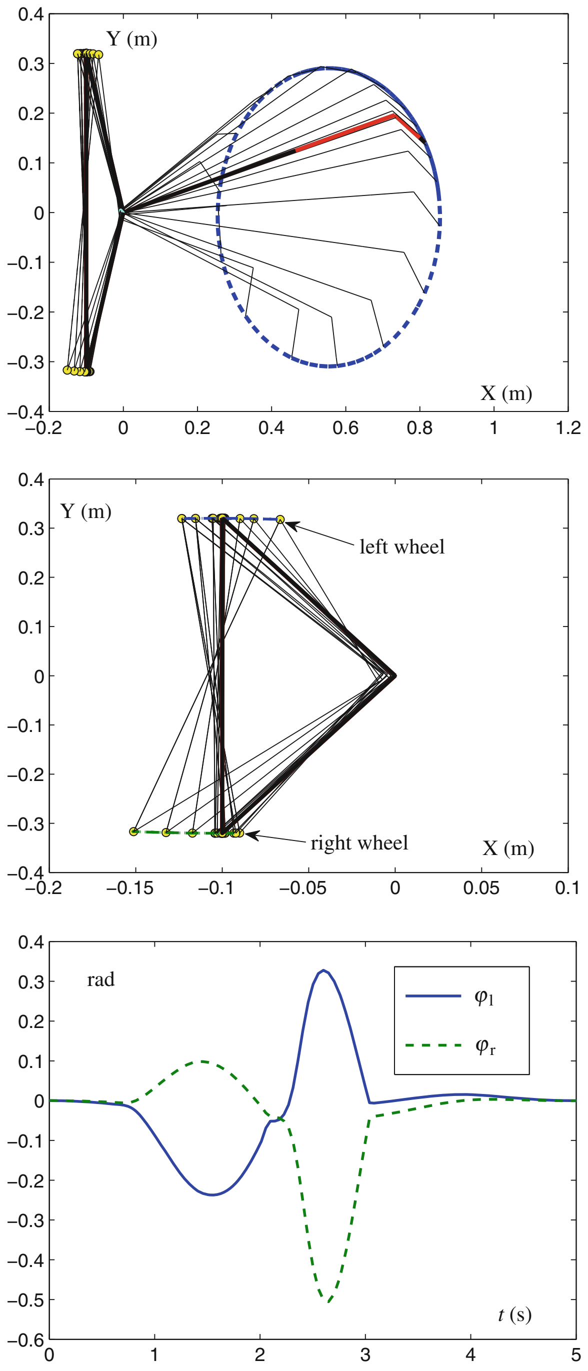


Fig. 15.5 More simulation results when the mobile robot manipulator tracks the given circular path synthesized by the proposed RMP scheme (15.11)–(15.14)

Repetitive motion To finish the circular-path tracking task in a repetitive manner, the proposed RMP scheme (15.11)–(15.14) with $\gamma_1 = \gamma_2 = \gamma_3 = 10^5$ and the discrete-time QP solver are applied to the control of the mobile robot manipulator. The corresponding simulation results are shown in Figs. 15.4, 15.5, 15.6, and 15.7.

Specifically, the upper graph of Fig. 15.4 shows the motion trajectories of the mobile robot manipulator during the whole tracking process. As seen from such a graph, the proposed RMP scheme (15.11)–(15.14) not only coordinates simultaneously the mobile platform and the manipulator to complete the given end-effector task, but also makes the mobile manipulator return to the initial state. In addition, from the middle graph of Fig. 15.4, we can see that the actual end-effector motion trajectory is close enough to the desired circular path. The lower graph of Fig. 15.4 shows the corresponding tracking position errors. As seen from such a graph, the corresponding X-axis, Y-axis, and Z-axis components of the tracking position error are less than 5×10^{-6} m. These results substantiate that the mobile robot manipulator

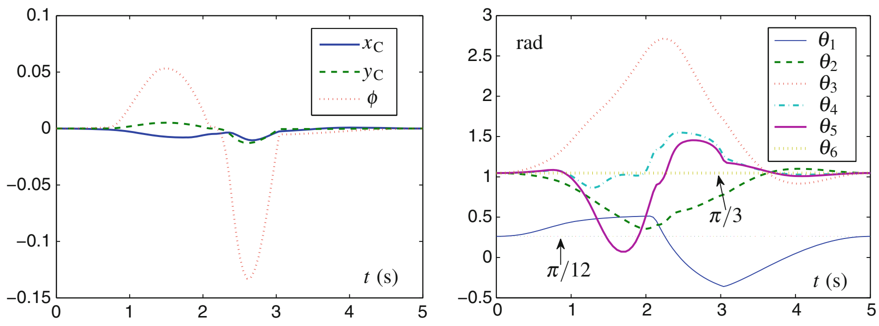


Fig. 15.6 Profiles of point of junction $[x_C, y_C]^T$, heading angle ϕ and joint angle θ when the mobile robot manipulator tracks the given circular path synthesized by the proposed RMP scheme (15.11)–(15.14)

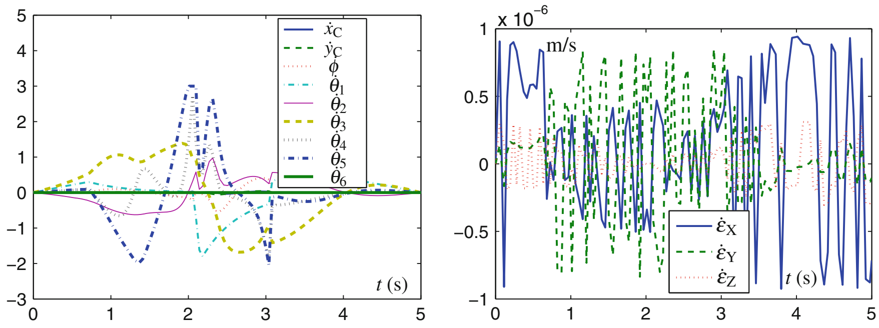


Fig. 15.7 Profiles of $[\dot{x}_C, \dot{y}_C]^T$, heading velocity $\dot{\phi}$, joint velocity $\dot{\theta}$ and the corresponding tracking velocity error when the mobile robot manipulator tracks the given circular path synthesized by the proposed RMP scheme (15.11)–(15.14)

can finish the circular path tracking task well, as synthesized by the proposed RMP scheme (15.11)–(15.14).

To further illustrate and verify the effectiveness of the proposed RMP scheme (15.11)–(15.14), in Fig. 15.5, we show the movements of the mobile platform. Specifically, the upper graph of Fig. 15.5 shows the top view of motion trajectories of the mobile robot manipulator, the middle graph of Fig. 15.5 shows the motion trajectories of the mobile platform, and the lower graph of Fig. 15.5 shows the corresponding left and right wheel profiles. From Fig. 15.5, we can conclude that the mobile platform is repetitive after the end-effector completing the circular-path tracking task.

Besides, what we are more interested in are some important variables, such as point of junction $[x_C, y_C]^T$, heading angle ϕ and joint angle θ. This reason lies in that if and only if these variables return to their initial states, the mobile manipulator can achieve the repetitive motion. Figure 15.6 shows the profiles of $[x_C, y_C]^T$, ϕ, and θ when the mobile manipulator tracks the given circular path. As seen from the left graph of Fig. 15.6, x_C, y_C, and ϕ return to their initial states. In addition, the right graph of Fig. 15.6 shows that θ also returns to its initial state $\theta(0)$. That is to say, these variables all return to their initial states, so that the mobile robot manipulator can achieve the repetitive motion. Figure 15.7 shows the profiles of $[\dot{x}_C, \dot{y}_C]^T$, heading velocity $\dot{\phi}$, joint velocity $\dot{\theta}$, and the corresponding tracking velocity error when the mobile manipulator tracks the given circular path. From Fig. 15.7, we can see that the final states of the combined velocity equal zero, and that the corresponding X-axis, Y-axis, and Z-axis components of the tracking velocity error are less 1×10^{-6} m. The results further substantiate the high accuracy of the proposed RMP scheme (15.11)–(15.14). It is worth pointing out that, in the left graph of Fig. 15.7, $\dot{\theta}_5$ reaches its upper bound $\dot{\theta}_5^+$, but never exceed the upper bound, which demonstrates that the bound constraint (15.14) is activated and effective.

15.4.2 Lissajous-Figure Path Tracking

In this subsection, the end-effector of the mobile robot manipulator is expected to track a Lissajous-figure path with parameter χ being 0.45 m. The X-axis, Y-axis, and Z-axis velocity functions of the desired Lissajous-figure path are

$$\begin{cases} \dot{r}_{\mathrm{dwX}}(t) = -\frac{4\pi^2\chi}{T}\sin(4\pi\sin^2(\frac{\pi t}{2T}) + \pi/6)\sin(\frac{\pi t}{2T})\cos(\frac{\pi t}{2T}), \\ \dot{r}_{\mathrm{dwY}}(t) = \frac{2\pi^2\chi}{T}\cos(2\pi\sin^2(\frac{\pi t}{2T}) + \pi/6)\sin(\frac{\pi t}{2T})\cos(\frac{\pi t}{2T}), \\ \dot{r}_{\mathrm{dwZ}}(t) = 0. \end{cases}$$

In order to investigate the effectiveness of the proposed RMP scheme (15.11)–(15.14) for different values of parameters, in this example, we set task duration $T = 10$ s, $\gamma_1 = \gamma_2 = 10^3$, and $\gamma_3 = 10^2$. Thus, the corresponding simulation results, synthesized by the proposed RMP scheme (15.11)–(15.14) and the discrete-time QP solver (15.19), are shown in Figs. 15.8, 15.9, 15.10, and 15.11.

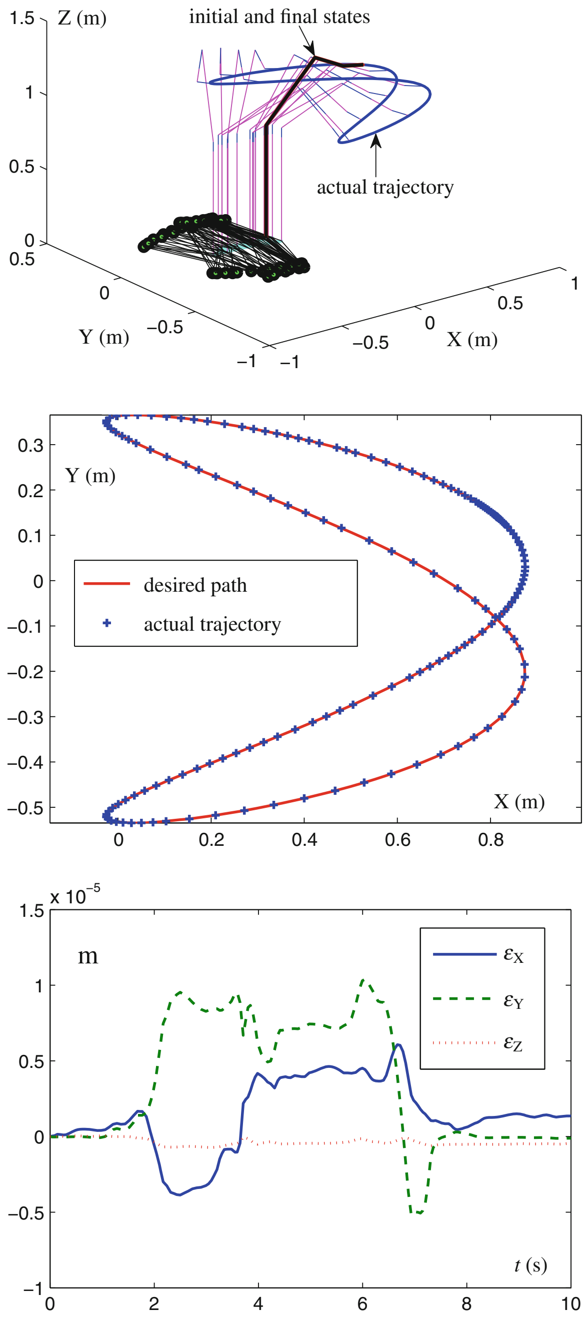


Fig. 15.8 Simulation results when the mobile robot manipulator tracks the given Lissajous-figure path synthesized by the proposed RMP scheme (15.11)–(15.14)

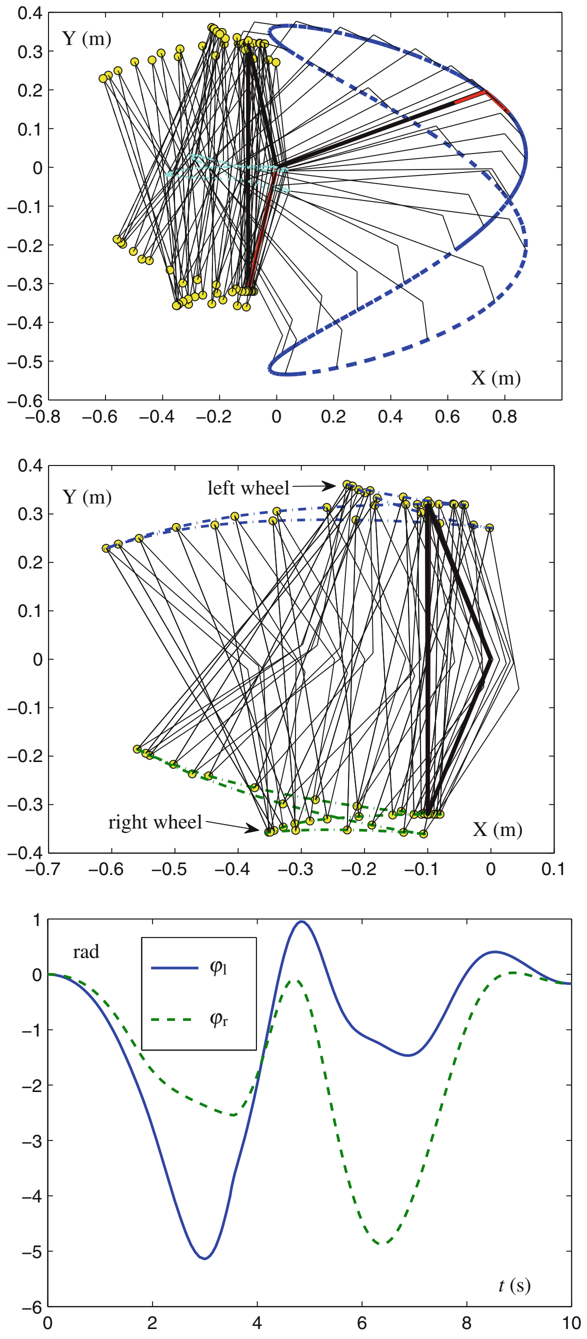


Fig. 15.9 More simulation results when the mobile robot manipulator tracks the given Lissajous-figure path synthesized by the proposed RMP scheme (15.11)–(15.14)

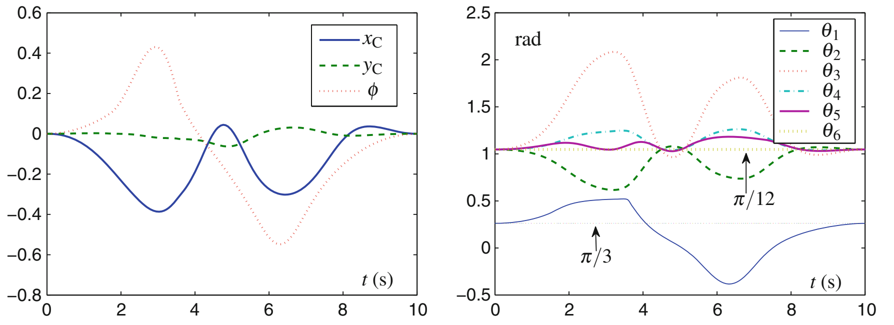


Fig. 15.10 Profiles of point of junction $[x_C, y_C]^T$, heading angle ϕ and joint angle θ when the mobile robot manipulator tracks the given Lissajous-figure path synthesized by the proposed RMP scheme (15.11)–(15.14)

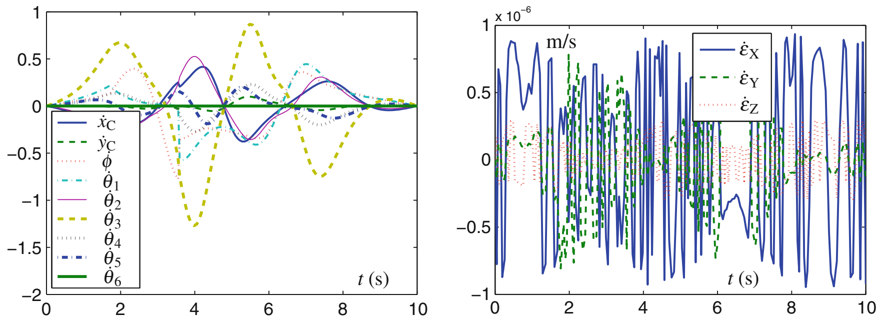


Fig. 15.11 Profiles of $[\dot{x}_C, \dot{y}_C]^T$, heading velocity $\dot{\phi}$, joint velocity $\dot{\theta}$, and the corresponding tracking velocity error when the mobile robot manipulator tracks the given Lissajous-figure path synthesized by the proposed RMP scheme (15.11)–(15.14)

The upper graph of Fig. 15.8 shows the whole tracking process of the mobile robot manipulator. As seen from such a graph, the RMP purpose of the mobile robot manipulator is achieved. Besides, the middle graph of Fig. 15.8 shows the desired Lissajous-figure path and the actual end-effector trajectory, and the lower graph of Fig. 15.8 shows the corresponding tracking position errors. As observed from the middle graph of Fig. 15.8, the actual motion trajectory of the mobile robot manipulator's end-effector is sufficiently close to the desired Lissajous-figure path. The corresponding X-axis, Y-axis, and Z-axis components of the tracking position error shown in the lower graph of Fig. 15.8 are less than 1×10^{-5} m. These demonstrate that the given Lissajous-figure path tracking task is performed well via the proposed RMP scheme (15.11)–(15.14).

To see more clearly the repetitive motion of the mobile robot manipulator, its mobile platform is visualized in Fig. 15.9. Specifically, the upper graph of Fig. 15.9 shows the top view of motion trajectories of the mobile robot manipulator, the middle graph of Fig. 15.9 shows the motion trajectories of the mobile platform, and the lower graph of Fig. 15.9 shows the corresponding left and right wheel profiles. It follows from Fig. 15.9 that the mobile platform is repetitive after the end-effector completing the Lissajous-figure path tracking task.

Figure 15.10 shows the profiles of $[x_{\mathrm{C}}, y_{\mathrm{C}}]^{\mathrm{T}}$, ϕ, and θ when the mobile manipulator tracks the given Lissajous path. As seen from the left graph of Fig. 15.10, x_{C}, y_{C}, and ϕ go back to their initial states. In addition, the right graph of Fig. 15.10 shows that θ also returns to its initial state $\theta(0)$. These illustrate and verify again the effectiveness of such a repetitive motion scheme and the non-repetitive problem of the mobile manipulator has been solved by using the repetitive motion scheme. Figure 15.11 further shows the profiles of $[\dot{x}_{\mathrm{C}}, \dot{y}_{\mathrm{C}}]^{\mathrm{T}}$, heading velocity $\dot{\phi}$, joint velocity $\dot{\theta}$ and the corresponding tracking velocity error when the mobile manipulator tracks the given Lissajous path. From the left graph of Fig. 15.11, we can see that the final states of the combined velocity equal zero. Note that, if the final states of combined velocity are not zero, the mobile manipulator will not stop immediately at the end of the task duration; and thus the non-repetitive problem may happen. From the right graph of Fig. 15.11, one can see that the corresponding X-axis, Y-axis, and Z-axis components of the tracking velocity error are less 1×10^{-6} m. The results further demonstrate the high accuracy of the proposed RMP scheme (15.11)–(15.14) and discrete-time QP solver (15.19). It is also worth noting that all the variables (e.g., θ, $\dot{\theta}$, ϕ, and φ) in simulations are kept within their limits due to consideration of physical constraints of mobile manipulators. Thus, the given end-effector task can be completed successfully.

In summary, the presented two examples performed on the wheeled mobile robot manipulator, i.e., tracking a circular path and a Lissajous-figure path, have both substantiated the efficacy of the proposed RMP scheme (15.11)–(15.14) and the corresponding discrete-time QP solver (15.19), which can solve the non-repetitive problem well. Furthermore, the tracking position and velocity errors shown in Figs. 15.4, 15.7, 15.8 and 15.11, have validated well the high accuracy of such a RMP scheme (15.11)–(15.14). Besides, these simulation results have shown again the successful application of the presented ZD approach to RMP of mobile redundant robot manipulators.

15.5 Summary

In this chapter, by defining three different ZFs and by exploiting the ZD design formula, the velocity-level RMP performance index (15.10) has been proposed, developed, and investigated. Based on such a performance index, the velocity-level RMP scheme (15.11)–(15.14) has been further presented and investigated for mobile redundant robot manipulators, which is reformulated as a QP (15.15)–(15.17) and then is solved by the numerical algorithm (15.19). Computer simulation results based on the wheeled mobile robot manipulator with different illustrative examples have substantiated well the effectiveness, accuracy, and safety of the proposed velocity-level

RMP scheme for physically-constrained mobile redundant robot manipulators, and more importantly, have shown once again the application prospect of the presented ZD approach to robotic redundancy resolution.

References

1. Lasky TA, Ravani B (2000) Sensor-based path planning and motion control for a robotic system for roadway crack sealing. IEEE Trans Control Syst Technol 8(4):609–622
2. Xia Y, Wang J (2001) A dual neural network for kinematic control of redundant robot manipulators. IEEE Trans Syst, Man, Cybern B 31(1):147–154
3. Shahri NR, Troch I (1996) Collision-avoidance for redundant robots through control of the self-motion of the manipulator. J Intell Robot Syst 16(2):123–149
4. Ding H, Chan SP (1996) A real-time planning algorithm for obstacle avoidance of redundant robots. J Intell Robot Syst 16(3):229–243
5. Zhang Y, Zhang Z (2013) Repetitive motion planning and control of redundant robot manipulators. Springer, New York
6. Guo D, Zhang Y (2014) Acceleration-level inequality-based MAN scheme for obstacle avoidance of redundant robot manipulators. IEEE Trans Ind Electron 61(12):6903–6914
7. Guo D, Zhang Y (2014) Simulation and experimental verification of weighted velocity and acceleration minimization for robotic redundancy resolution. IEEE Trans Autom Sci Eng 11(4):1203–1217
8. Guo D, Zhang Y (2014) Li-function activated ZNN with finite-time convergence applied to redundant-manipulator kinematic control via time-varying Jacobian matrix pseudoinversion. Appl Soft Comput 24:158–168
9. Klein CA, Kee KB (1989) The nature of drift in pseudoinverse control of kinematically redundant manipulators. IEEE Trans Robot Autom 5(2):231–234
10. Klein CA, Ahmed S (1995) Repeatable pseudoinverse control for planar kinematically redundant manipulators. IEEE Trans Syst, Man, Cybern 25(12):1657–1662
11. Siciliano B, Khatib O (2008) Springer handbook of robotics. Springer, Heidelberg
12. Siciliano B, Sciavicco L, Villani L, Oriolo G (2009) Robotics: modelling, planning and control. Springer, London
13. Zhang Z, Zhang Y (2012) Acceleration-level cyclic-motion generation of constrained redundant robots tracking different paths. IEEE Trans Syst Man Cybern B 42(4):1257–1269
14. Zhang Y, Tan Z, Chen K, Yang Z, Lv X (2009) Repetitive motion of redundant robots planned by three kinds of recurrent neural networks and illustrated with a four-link planar manipulator's straight-line example. Robot Auton Syst 57(6–7):645–651
15. Zhang Y, Lv X, Li Z, Yang Z, Chen K (2008) Repetitive motion planning of PA10 robot arm subject to joint physical limits and using LVI-based primal-dual neural network. Mechatronics 18(9):475–485
16. Shamir T, Yomdin Y (1988) Repeatability of redundant manipulators: Mathematical solution of the problem. IEEE Trans Autom Control 33(6):1004–1009
17. Roberts RG, Maciejewski AA (1993) Repeatable generalized inverse control strategies for kinematically redundant manipulators. IEEE Trans Autom Control 38(5):689–698
18. Tchon K, Jakubiak J (2005) A repeatable inverse kinematics algorithm with linear invariant subspaces for mobile manipulators. IEEE Trans Syst, Man, Cybern B 35(5):1051–1057
19. Lee JK, Cho HS (1997) Mobile manipulator motion planning for multiple tasks using global optimization approach. J Intell Robot Syst 18(2):169–190
20. Bayle B, Fourquet J-Y,M. Renaud M (2003) Manipulability of wheeled mobile manipulators: Application to motion generation. Int J Robot Res 22(7–8):565–581

21. Xu D, Zhao D, Yi J, Tan X (2009) Trajectory tracking control of omnidirectional wheeled mobile manipulators: Robust neural networkbased sliding mode approach. IEEE Trans Syst, Man, Cybern B 39(3):788–799
22. Xiao L, Zhang Y (2014) A new performance index for the repetitive motion of mobile manipulators. IEEE Trans Cybern 44(2):280–292
23. Tchoń K (2002) Repeatability of inverse kinematics algorithms for mobile manipulators. IEEE Trans Autom Control 47(8):1376–1380
24. Tchoń K, Jakubiak J (2003) Endogenous configuration space approach to mobile manipulators: a derivation and performance assessment of Jacobian inverse kinematics algorithms. Int J Control 76:1387–1419
25. Tchoń K, Jakubiak J (2005) A repeatable inverse kinematics algorithm with linear invariant subspaces for mobile manipulators. IEEE Trans Syst, Man, Cybern B 35(5):1051–1057
26. Tchoń K (2006) Repeatable, extended Jacobian inverse kinematics algorithm for mobile manipulators. Syst Control Lett 55:87–93
27. Zhang Y, Yi C (2011) Zhang neural networks and neural-dynamic method. Nova Science Publishers, New York
28. Tang C, Miller P, Krovi V, Ryu J, Agrawal S (2011) Differential-flatness-based planning and control of a wheeled mobile manipulator-Theory and experiment. IEEE/ASME Trans Mechatron 16(4):768–773
29. White GD, Bhatt RM, Tang CP, Krovi VN (2009) Experimental evaluation of dynamic redundancy resolution in a nonholonomic wheeled mobile manipulator. IEEE/ASME Trans Mechatron 14(3):349–357
30. He B (1994) Solving a class of linear projection equations. Numer Math 68(1):71–80
31. He B (1994) A new method for a class of linear variational inequalities. Math Program 66(2):137–144
32. Guo D, Zhang Y (2012) A new inequality-based obstacle-avoidance MVN scheme and its application to redundant robot manipulators. IEEE Trans Syst, Man, Cybern, C 42(6):1326–1340