

Long Jin
Lin Wei
Xin Lv

Neural Dynamics for Time-varying Problems

Advances and Applications

Neural Dynamics for Time-varying Problems

Long Jin • Lin Wei • Xin Lv

Neural Dynamics for Time-varying Problems

Advances and Applications



Springer

Long Jin
School of Information Science and
Engineering
Lanzhou University
Lanzhou, China

Lin Wei
School of Information Science and
Engineering
Lanzhou University
Lanzhou, China

Xin Lv
School of Information Science and
Engineering
Lanzhou University
Lanzhou, China

ISBN 978-3-031-68593-4 ISBN 978-3-031-68594-1 (eBook)
<https://doi.org/10.1007/978-3-031-68594-1>

© The Editor(s) (if applicable) and The Author(s), under exclusive license to Springer Nature Switzerland AG 2025

This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

If disposing of this product, please recycle the paper.

To our ancestors and parents, as always

Preface

In the past decades, neural dynamics has received considerable studies in many scientific and engineering fields. As a powerful online computing tool with potential application scenarios, neural dynamics is becoming an independent research direction in online computational field. Remarkable advances have been made in the area of neural dynamics for online computing. For example, a constrained optimization problem with time-varying objective functions and constraints is referred as time-varying constrained optimization problem. Early works often solve it through removing the explicit constraints with a penalty term of the objective function and then designing a neural dynamics model evolving along the gradient descent direction by leveraging the short-time invariance. This type of neural dynamics only converges to an approximation of the optimal solution. Many complex problems can be modeled as optimizations or nonlinear equations or even tensor equations, and then, traditionally, gradient descent or projected gradient descent is employed to get the corresponding dynamic equations for online solutions to these problems.

In this book, we present the design, proposal, development, analysis, modeling, and simulation of various neural dynamic models, along with their respective applications including motion planning of redundant manipulators, filter design, winner-take-all operation, multiple-input multiple-output system configuration, multi-linear tensor equation solving, and manipulability optimization. Specifically, starting from the top-level considerations of hardware implementation, we integrate computational intelligence methods and control theory to design a series of dynamic and noise-resistant discrete neural dynamic methods. The research work not only owns the theoretical guarantee on its convergence, noise resistance, and accuracy, but demonstrates the effectiveness and robustness in handling various optimization and equation solving problems with time-varying properties, particularly in handling time-varying problems and noise perturbations. Moreover, by reducing complexity and avoiding matrix inversion operations, the models' feasibility and practicality are further enhanced.

The idea for this book on constructing neural dynamics models was conceived during classroom teaching, the research discussion in the laboratory, and international scientific meetings. All of the materials of this book are derived

from the authors' papers published in journals, such as *IEEE Transactions on Neural Networks and Learning Systems*, *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *IEEE Transactions on Industrial Informatics*, and *IEEE Transactions on Emerging Topics in Computational Intelligence*. In fact, since the early 2010s, the field of neural dynamics has undergone the phases of exponential growth, generating many new theoretical concepts and tools (including the authors' ones). At the same time, these theoretical results have been successfully applied to the solution of many practical problems. Our priority is thus to cover each central topic in enough details 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 is classified into the following seven chapters.

Chapter 1—In this chapter, we build a systematic framework encompassing existing neural dynamics models for solving QP and analyzes the intrinsic similarity of the models by casting the diversity of model expression, derivation, and structure design aside. Then, this work makes progress along this direction by applying control-theoretical techniques to construct new recurrent neural dynamics for manipulating a perturbed nonstationary QP with time-varying parameters considered. The normalization of neural dynamics methods for QP optimization is first performed. The classification and superior technology of models/controllers are then discussed at the control level. Specifically, to break the limitations of existing continuous-time models in handling nonstationary problems, a discrete recurrent neural dynamics model is proposed to robustly deal with noise. In addition, we conduct computer simulations and prospective applications to substantiate the efficacy and advantages of the proposed neural dynamics model for solving nonstationary QPs in the presence of perturbation.

Chapter 2—In this chapter, different from the model presented in Chap. 1, we present two discrete-time neural dynamics models for solving the online time-dependent complex-valued QP problems subject to linear constraints, which can achieve global convergence performance in the presence of perturbations with the provided theoretical analyses. In addition, the second proposed model is developed to eliminate the operation of explicit matrix inversion by introducing the quasi-Newton Broyden-Fletcher-Goldfarb-Shanno (BFGS) method. Computer simulation results and applications in robotics and filters are provided to illustrate the feasibility and superiority of the proposed models in comparison with the existing solutions.

Chapter 3—In this chapter, to further investigate future dynamic nonlinear optimization (FDNO) problems with time-varying parameters, a noise-tolerant neural dynamics (NTND) model is proposed and investigated, breaking the short-time invariance hypothesis of various existing methods and eliminating the large lagging error. The proposed NTND model is further developed to reduce computational burden and eliminate the need for matrix inversion. Theoretical analyses demonstrate global convergence, and numerical experiments confirm the superiority of the proposed algorithms for online solutions of FDNO problems.

Chapter 4—In this chapter, a noise-suppressing discrete-time neural dynamics (NSDTND) model is proposed for solving the time-variant nonlinear optimization with equality and inequality constraints (TVNOEIC) problem. Theoretical analysis

is presented to prove that the residual error of the proposed model is tiny enough to estimate the actual solution even in the presence of noise perturbation. In addition, numerical simulations, including an application on winner-take-all operation, are provided to further demonstrate the effectiveness and robustness of the proposed NSDTND model.

Chapter 5—In this chapter, a neural dynamics model is constructed and investigated for solving time-dependent Sylvester equation with matrix inversion involved in the solving process. Besides, to eliminate the matrix inversion in the model, the quasi-Newton BFGS method is leveraged. Moreover, the global convergence performance and the effectiveness of the two discrete computational models are testified by providing theoretical analyses and numerical experiments with comparisons to the existing solutions, respectively. Two applications to robotics and the multiple-input multiple-output system are given to elucidate the feasibility of the proposed models for solving time-dependent Sylvester equation.

Chapter 6—In this chapter, a high-order robust discrete-time neural dynamics (HRDND) model with a directly discrete approach is proposed for solving the time-varying multi-linear tensor equation (TMTE). Theoretical analyses on convergence and robustness are provided to prove that the proposed HRDND method is feasible and effective. Simulative experiments on four time-varying numerical examples and an application derived from the Bellman equation solved by the proposed HRDND method and other four methods are given, whose results illustrate the superiority of the proposed HRDND method.

Chapter 7—In this chapter, a collaborative neural dynamics (CND) model that fuses the advantages of evolutionary computation and neural dynamics methods is proposed for solving time-varying nonconvex optimization problems, which follows a meta-heuristic rule and exploits gradient-based neural solution with robustness (GNSR) to deal with different noises. The convergence and robustness of CND model and GNSR model are demonstrated by the theoretical analyses. Simulative comparisons with existing methods and an application to manipulability optimization on a redundant manipulator with physical experiments substantiate the superiority of the proposed models in solving the nonconvex time-varying optimization problems.

In summary, this book presents different kinds of neural dynamics models with variant contributions, and further applies these models to diverse scenarios. 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 neural dynamics models as well as their applications is provided, which opens an avenue and may promise to become a major inspiration

for studies and researches in neural dynamics, robotics, and tensor. Without a doubt, this book can be extended. Any comments or suggestions are welcome. The authors can be contacted via e-mail: jinlong@lzu.edu.cn.

Lanzhou, China
May 2023

Long Jin

Acknowledgments

During the work on this book, we have had the pleasure of discussing its various aspects and results with many cooperators and students, including Xin Zheng, Huanmei Wu, and Xiaoyan Zhang. We highly appreciate their contributions, which particularly allowed us to improve our book manuscript.

The continuous support of our research by the National Natural Science Foundation of China (with number 62176109) 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

1	Neural Dynamics Based on Control Theoretical Techniques	1
1.1	Introduction	1
1.2	Scheme Formulation and Neural Dynamics Solution	3
1.2.1	Preliminaries and Problem Formulation	3
1.2.2	Neural Dynamics Solutions	5
1.2.3	Compact Reformulation	9
1.3	Theoretical Verifications	11
1.4	Illustrative Examples	21
1.4.1	Example 1	21
1.4.2	Example 2 (Application in Robots)	24
1.5	Extension of the Bounded QP	32
1.5.1	Preliminaries	33
1.5.2	Application in κ -WTA	34
1.6	Conclusion	36
	References	36
2	Complex-Valued Discrete-Time Neural Dynamics	39
2.1	Introduction	39
2.2	Preliminary and Scheme Formulation	41
2.3	Discrete-Time Neural Dynamics Models	43
2.3.1	Complex-Valued DTND Model	43
2.3.2	Complex-Valued DTND-BFGS Model	45
2.3.3	Model Simplification	46
2.4	Theoretical Analyses and Verification	47
2.5	Numerical Experiments	53
2.6	Applications	63
2.6.1	Applications to Robotics	63
2.6.2	Applications to Complex-Valued FIR Filter Design	64
2.7	Summary	68
	References	68

3	Noise-Tolerant Neural Dynamics	71
3.1	Introduction	71
3.2	Problem Formulation and Solutions	73
3.2.1	Problem Formulation	73
3.2.2	Continuous-Time NTND Model	73
3.2.3	Existing Discrete-Time Solutions	75
3.2.4	NTND and NTND-BFGS Models	76
3.3	Theoretical Analyses and Results	78
3.4	Numerical Experiments	85
3.4.1	Example 1	85
3.5	Application to Motion Generation	91
3.6	Conclusions	95
	References	96
4	Computational Neural Dynamics	99
4.1	Introduction	99
4.2	Problem Formulation and NSDTND Model	101
4.2.1	Problem Formulation	102
4.2.2	Continuous-Time Neural Dynamics Model	103
4.2.3	NSDTND Model	103
4.2.4	Models for Comparison	104
4.3	Theoretical Analysis and Results	104
4.3.1	Precondition	105
4.3.2	Convergence of NSDTND Model	106
4.3.3	Convergence Under Noisy Condition	108
4.4	Illustrative Examples	111
4.4.1	TVNOEIC Problem-Solving	111
4.4.2	Comparisons	114
4.4.3	WTA Application	114
4.5	Conclusions	118
	References	118
5	Discrete Computational Neural Dynamics	121
5.1	Introduction	121
5.2	Problem Formulation	122
5.2.1	Redundant Robot Manipulator	123
5.2.2	Description of the DCND Model	123
5.2.3	Description of the DCND-BFGS Model	124
5.2.4	The DCND Model Simplification	125
5.3	Control Theoretical Analyses	127
5.4	Illustrative Examples	132
5.4.1	Example 1	132
5.4.2	Example 2: Tracking Control of Robotics	139
5.4.3	Example 3: Linear Detection for MIMO	139
5.5	Conclusions	141
	References	141

6	High-Order Robust Discrete-Time Neural Dynamics	145
6.1	Introduction	145
6.2	Problem Formulation and Solutions	148
6.2.1	Problem Formulation	148
6.2.2	HRDND Method	150
6.2.3	SOTA Methods	151
6.3	Theoretical Analyses	152
6.3.1	Nonlinear Transformation	152
6.3.2	Convergence Analyses	154
6.3.3	Robustness Analyses	156
6.4	Simulation Verifications	159
6.4.1	Example 1	159
6.4.2	Example 2	159
6.4.3	Application	161
6.5	Conclusion	163
	References	165
7	Collaborative Neural Dynamics	167
7.1	Introduction	167
7.2	Problem Formulation and Model Description	169
7.2.1	Problem Formulation	169
7.2.2	Collaborative Neural Dynamics	170
7.3	Theoretical Analyses and Results	172
7.4	Simulation Verification	178
7.4.1	Example I	179
7.4.2	Example II	183
7.4.3	Simulation Comparisons	184
7.5	Application to Manipulability Optimization of Redundant Manipulator	193
7.6	Conclusions	194
7.7	Appendix A	194
7.8	Appendix B	195
	References	197
	Glossary	201

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	Zeroing dynamics

Chapter 1

Neural Dynamics Based on Control Theoretical Techniques



1.1 Introduction

It has been pointed out that the nature of solving equations in computation is closely related to the control of dynamic systems [1, 2], with the corresponding residual errors required to approach near zero robustly, accurately, and rapidly. Based on such a relationship, efforts have been devoted to revisiting, designing, and analyzing computational methods by leveraging control theoretical techniques [1, 3, 4]. A typical way to leverage the control technology in solving algebraic problems is to select a nonnegative error function, of which the minimum value corresponds to the equilibrium point of the original task problem [5–7]. Then, designing a suitable controller (i.e., the designed computational model) based on a Lyapunov theory leads to the convergence of the residual error, and thus, the task function is solved. For example, the control Lyapunov function approach in [1, 3] derives several iterative methods by viewing them as dynamic systems with feedback control, which is often used in the design and analysis of numerical methods. The methods (systems) possessed with feedback control are classified into the recursive ones, which recursively estimate a solution based on the error-feedback information generated by the previous data for achieving the given accuracy. By contrast, the direct methods achieve the theoretical solution in manners of error-free and finite steps, but are not able to solve complex problems effectively.

The recurrent neural dynamic solvers, as parallel computing methods, are now considered as a powerful tool for solving real-time optimization problems with superior performance in large-scale applications [8–10]. In control terminology, the existing computing methods are different in the formula formations, derivation approaches, and construction techniques, but share a common characteristic in finding the evolving direction [11], of which each one is equivalent to a feedback/proportional controller with different (varying) feedback gain. In particular, a fixed-point iteration and Newton's method for iteratively solving nonlinear equations in a control theoretical framework are investigated in [12]. Furthermore,

numerical iterative algorithms in finite precision are considered as dynamic systems using control theory in [13]. Similarly, gradient-based neural dynamics models regarded as feedback controllers are usually applied to cope with the time-invariant convex quadratic programming (QP) [14, 15].

It is a common view in the control society that the lagging error is unavoidable for a dynamic system aided with a feedback/proportional controller, thereby leading the residual errors not to converge to zero in the limited time with the controllers out of order [16]. Additionally, it is difficult to design a computing method for handling nonstationary problems from the computing framework, whereas many techniques for handling such knotty problems are available in control fields. An intuitive method is to exploit the time-derivation information for predicting the evolving direction. By exploiting this technique, a gradient-based searching method is considered to handle a class of distributed nonstationary quadratic optimization problems with the errors asymptotically stable and uniformly controlled within an arbitrarily small value [17]. To further improve the efficiency of the controllers, a varying-parameter convergent-differential neural network is designed in [18] with different activation functions. In [19], a regularized and smoothed Fischer-Burmeister method is applied in the Newton method for solving nonlinear convex quadratic programs in real time with derivative information involved.

However, the existing frameworks on computation, such as numerical methods, essentially lack a mechanism to directly consider noises, which have negative impacts on the accuracy of the calculations or the stability of the controllers. In contrast, the control perspective has tools, such as integral control and internal model principle [20], to deal with noises. The shift from computation perspective to control perspective stimulates us to leverage the unique control tools for performance improvement. In addition, time is precious for solving nonstationary problems in practice. Therefore, it is worth exploring neural dynamics via a systematic approach from the control perspective for solving the nonstationary QP with the inherent noise-suppression property. In this chapter, an error-sum enhanced neural dynamics model is first constructed with integral control embedded. Then, considering that the traditional methods (Newton iteration, the gradient-based neural dynamics model) are defective in dealing with nonstationary problems and eliminating noises, the time-derivative information and error-integration information are incorporated to build the two modified ones, which improves the computing performance and the stability of the controllers.

From the viewpoint of control, this chapter builds a systematic framework that encompasses the existing algorithms/models for solving QP and first analyzes the intrinsic similarity of the models by casting the diversity of model expression, the derivation, and structure design aside. For one thing, the normalization of the existing models is completed. For another, the classification and the superior technology of models/controllers are discussed at the control level. Overall, these three presented models are designed aided with the systemic framework in the control terminology for nonstationary problems solving with robustness and effectiveness.

1.2 Scheme Formulation and Neural Dynamics Solution

This section provides the preliminaries and problem formulation on the nonstationary QP with the new model presented.

1.2.1 Preliminaries and Problem Formulation

A nonstationary QP problem is depicted in the form of

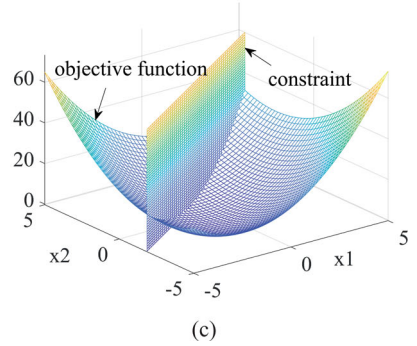
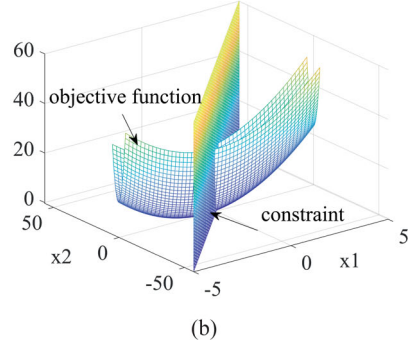
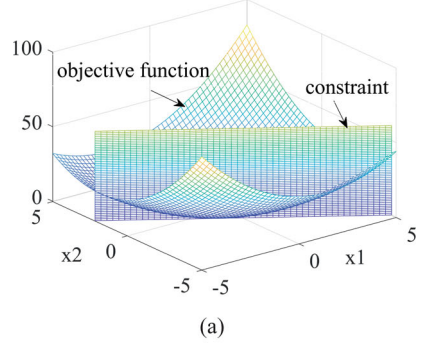
$$\begin{aligned} \text{Minimize} \quad & \mathbf{x}^T(t)H(t)\mathbf{x}(t)/2 + \boldsymbol{\rho}^T(t)\mathbf{x}(t) \\ \text{Subject to} \quad & \Lambda(t)\mathbf{x}(t) = \mathbf{v}(t), \end{aligned} \quad (1.1)$$

where superscript T is used to denote the transpose operation; $\mathbf{x}(t) \in \mathbb{R}^n$ represents the unknown vector to be computed; Hessian matrix $H(t) \in \mathbb{R}^{n \times n}$ is assumed to be positive definite in this work; parameters $\boldsymbol{\rho}(t) \in \mathbb{R}^n$, $\Lambda(t) \in \mathbb{R}^{m \times n}$, $\mathbf{v}(t) \in \mathbb{R}^m$. A typical example is visualized in Fig. 1.1, from which one can observe that the linear-constraint plane $\Lambda(t)\mathbf{x}(t) = \mathbf{v}(t)$ and the objective function surface $\mathbf{x}^T(t)H(t)\mathbf{x}(t)/2 + \boldsymbol{\rho}^T(t)\mathbf{x}(t)$ are both varying with time t . In this regard, solving nonstationary QP problem (1.1) is of difficulty due to the fact that the optimal solution is nonstationary and varying with time t . With g denoting the sampling gap and subscript k being the update index, at the time instant $t = kg$, any algorithm aiming at solving (1.1) needs to input the values of the related parameters, e.g., $H(kg)$, and then output the computed $\mathbf{x}(kg)$ after conducting the computing operations. However, computing consumes time inevitably, which leads to the lagging errors evidently. By its very nature, there exists a contradiction in handling dynamic problems. That is, the solution $\mathbf{x}((k+1)g)$ should be computed before the time instant $t = (k+1)g$ but the data $H((k+1)g)$, $\boldsymbol{\rho}((k+1)g)$, $\Lambda((k+1)g)$ and $\mathbf{v}((k+1)g)$ are not available until the time instant $t = (k+1)g$. The traditional approach for resolving such a contradiction is to exploit the hypothesis of short-time invariance, of which the computed result is directly outputted for the next instant. However, a high computational accuracy requires the frequent update of information on the parameters, leading to significant overhead in real-time computing. From the perspective of control, lacking the compensation for the velocity component of dynamic parameters, models and algorithms by leveraging the hypothesis of short-time invariance may fail to get the desired accuracy when the parameters vary too fast, not to mention more challenging situations such as handling noises or disturbances.

By leveraging the Lagrangian function [21], the nonstationary QP problem depicted in (1.1) can be converted into the following equation:

$$\Theta(t)\mathbf{y}(t) = \boldsymbol{\eta}(t), \quad (1.2)$$

Fig. 1.1 Three-dimensional snapshots of quadratic function, linear constraint, and optimal solution of a nonstationary QP problem at different time instants, with $H(t) = [\sin(t)/4 + 1 \quad \cos(t); 0 \quad \cos(t)/4 + 1]$, $\rho(t) = [\sin(3t); \cos(3t)]$, $\Lambda(t) = [\sin(4t); \cos(4t)]^T$, and $v(t) = \cos(2t)$. **(a)** Snapshot at time instant $t = 0.6$ s, **(b)** snapshot at time instant $t = 4.8$ s, and **(c)** snapshot at time instant $t = 8.85$ s



where

$$\Theta(t) = \begin{bmatrix} H(t) & \Lambda^T(t) \\ \Lambda(t) & 0 \end{bmatrix} \in \mathbb{R}^{(n+m) \times (n+m)},$$

$$\mathbf{y}(t) = \begin{bmatrix} \mathbf{x}(t) \\ \boldsymbol{\chi}(t) \end{bmatrix} \in \mathbb{R}^{n+m}, \quad \boldsymbol{\eta}(t) = \begin{bmatrix} -\boldsymbol{\rho}(t) \\ \mathbf{v}(t) \end{bmatrix} \in \mathbb{R}^{n+m},$$

with $\chi(t)$ denoting the Lagrange-multiplier vector. Many neural dynamics models and numerical algorithms have been widely reported for solving the nonstationary QP problem (1.1). For instance, the classical Newton iteration for solving (1.1) can be rewritten as

$$\mathbf{y}_{k+1} = \mathbf{y}_k - \Theta_k^{-1}(\Theta_k \mathbf{y}_k - \boldsymbol{\eta}_k) = \Theta_k^{-1} \boldsymbol{\eta}_k, \quad (1.3)$$

of which superscript $^{-1}$ represents the inverse operation of a matrix. In addition, the gradient-based neural network for solving the nonstationary QP problem (1.1) can be formulated as

$$\dot{\mathbf{y}}(t) = -\hat{\gamma} \Theta^T(t) (\Theta(t) \mathbf{y}(t) - \boldsymbol{\eta}(t)), \quad (1.4)$$

where $\hat{\gamma} > 0$ stands for the step size. It can be interpreted from (1.3) or (1.4) that the term $\Theta_k \mathbf{y}_k - \boldsymbol{\eta}_k$ or $\Theta(t) \mathbf{y}(t) - \boldsymbol{\eta}(t)$, which carries the residual error information, is leveraged to amend $\mathbf{y}_k$ for approximating $\mathbf{y}_{k+1}$. Evidently, both (1.3) and (1.4) can be placed in a category of the feedback controller in the control terminology, of which the residual error can be arbitrarily tiny for a small enough sampling gap g regarding (1.3) (or for a large enough design parameter $\hat{\gamma}$ regarding (1.4)).

1.2.2 Neural Dynamics Solutions

Traditionally, the following error function can be designed to monitor the online solution process of the nonstationary QP problem (1.1):

$$\boldsymbol{\varepsilon}(t) = \Theta(t) \mathbf{y}(t) - \boldsymbol{\eta}(t). \quad (1.5)$$

To ensure that $\mathbf{y}(t)$ is correspondingly driven to the desired solution $\mathbf{y}^*(t)$ of (1.2), the error $\boldsymbol{\varepsilon}(t)$, interpreted as a distance measure between $\mathbf{y}(t)$ and $\mathbf{y}^*(t)$, approaches zero with time. From the viewpoint of control, let $\mathbf{y}(t)$ be a state variable; $\mathbf{u}(t)$ a control input; $\boldsymbol{\varepsilon}(t)$ an output. A nonlinear dynamic system in control conventions can be written as follows:

$$d\mathbf{y}(t)/dt = \mathbf{u}(t)$$

$$\mathbf{z}(t) = \Theta(t) \mathbf{y}(t) - \boldsymbol{\eta}(t) = \boldsymbol{\varepsilon}(t). \quad (1.6)$$

The zero-finding problem of $\boldsymbol{\varepsilon}(t)$ can thus be described in the control terminology as finding a controller $\mathbf{u}(t)$ that drives the output $\boldsymbol{\varepsilon}(t)$ to zero and thus the state variable $\mathbf{y}(t)$ to the desired solution $\mathbf{y}^*(t)$.

Based on the design formula $\dot{\boldsymbol{\varepsilon}}(t) = -\gamma \boldsymbol{\varepsilon}(t)$, a neural dynamics model is presented in [21] and formulated as

$$\Theta(t)\dot{\mathbf{y}}(t) = -\dot{\Theta}(t)\mathbf{y}(t) + \dot{\boldsymbol{\eta}}(t) - \gamma(\Theta(t)\mathbf{y}(t) - \boldsymbol{\eta}(t)). \quad (1.7)$$

In addition, the continuous-time noise-tolerant design formula presented in [22] is written as

$$\dot{\boldsymbol{\epsilon}}(t) = -\gamma\boldsymbol{\epsilon}(t) - \lambda \int_0^t \boldsymbol{\epsilon}(\delta)d\delta. \quad (1.8)$$

Expanding the above design formula leads to

$$\begin{aligned} \Theta(t)\dot{\mathbf{y}}(t) = & -\dot{\Theta}(t)\mathbf{y}(t) + \dot{\boldsymbol{\eta}}(t) - \gamma(\Theta(t)\mathbf{y}(t) - \boldsymbol{\eta}(t)) \\ & - \lambda \int_0^t (\Theta(\delta)\mathbf{y}(\delta) - \boldsymbol{\eta}(\delta))d\delta. \end{aligned} \quad (1.9)$$

Hereto, to implement this continuous-time model on a digital computer, various numerical differentiation methods deriving from numerical interpolation formulas have been investigated, employed, and reported. However, most of them, from the perspective of data location, are unsuitable or even infeasible for handling nonstationary problems. For example, the Runge-Kutta methods are often leveraged in MATLAB tool “ode15s” to calculate $\mathbf{y}(t)$ from an ordinary differential equation depicted in the form of $\dot{\mathbf{y}}(t) = f(\mathbf{y}(t), t)$, which not only exploit the existing information but also rely on the future information that is not available during the computation interval. Evidently, this is because the data involving in the Runge-Kutta methods are located on both sides of the target point. To a certain extent, all of the existing continuous-time models for solving nonstationary problems implemented on a digital computer with future information exploited can be cast into the category of noncausal systems. Therefore, a suitable numerical differentiation formula for the discretization of continuous-time models is the key point to generate a computing method in compliance with the law of causality.

To this end, several numerical difference formulas have been constructed and analyzed to the discretization of continuous-time computing models specially designed for solving time-dependent problems in recent years [8, 23]. All of them share the same requirement in the location of the data points, i.e., there is one and only one point ahead of the desired target point. The Euler forward difference formula, as a typical method with simple structure, is able to conduct the discretization perfectly, which is shown as

$$\dot{\mathbf{y}}_k = (\mathbf{y}_{k+1} - \mathbf{y}_k)/g. \quad (1.10)$$

To further improve the solution accuracy, different difference formulas involving more data points are presented. For example, a five-step forward difference formula constructed in [8] is defined as

$$\dot{\mathbf{y}}_k = (24\mathbf{y}_{k+1} - 5\mathbf{y}_k - 12\mathbf{y}_{k-1} - 6\mathbf{y}_{k-2})/g$$

$$-4\mathbf{y}_{k-3} + 3\mathbf{y}_{k-4})/(48g). \quad (1.11)$$

It has been investigated in [8] that a discrete-time model based on (1.11) may achieve an $O(g^4)$ pattern in the residual error under the assumption that it is free of noise during the computations as well as the input data. In addition, from the perspective of control, any discrete-time model can be described as a time-delay system and more data points involved would inevitably result in less possibility in keeping stability for the derived model, raising the requirement on step size. Therefore, the classical Euler forward difference formula (1.10) is used to construct the discrete-time neural dynamics model in the ensuing part in consideration of the following facts. (1) Time is precious for nonstationary QP problems and thus the corresponding neural dynamics model should be of simple structure. (2) Noise pollutions are inevitable during the computing process, e.g., the roundoff errors and the truncation errors, and thus the corresponding neural dynamics model should be of high robustness and would be better if it is able to suppress or eliminate the noises. (3) The corresponding neural dynamics model should be of small time delay.

Therefore, using Euler forward difference formula (1.10) to discretize (1.9) generates

$$\begin{aligned} \mathbf{y}_{k+1} = & \mathbf{y}_k + g\Theta_k^{-1}(-\dot{\Theta}_k\mathbf{y}_k + \dot{\boldsymbol{\eta}}_k - \gamma(\Theta_k\mathbf{y}_k - \boldsymbol{\eta}_k) \\ & - \lambda g \sum_{j=0}^k (\Theta_k\mathbf{y}_k - \boldsymbol{\eta}_k)). \end{aligned} \quad (1.12)$$

By considering that the exact value of the time derivatives of Θ_k and $\boldsymbol{\eta}_k$ may be difficult to obtain, the Euler backward difference formula is applied to approximate them. Then, one has

$$\begin{aligned} \mathbf{y}_{k+1} = & \mathbf{y}_k + \Theta_k^{-1}((\Theta_{k-1} - \Theta_k)\mathbf{y}_k + \boldsymbol{\eta}_k - \boldsymbol{\eta}_{k-1} \\ & - \alpha(\Theta_k\mathbf{y}_k - \boldsymbol{\eta}_k) - \beta \sum_{j=0}^k (\Theta_k\mathbf{y}_k - \boldsymbol{\eta}_k)), \end{aligned} \quad (1.13)$$

where $\alpha = g\gamma$ and $\beta = g^2\lambda$. The above equation is termed the error-sum enhanced neural dynamics (ESEND) model. Similarly, the nonlinear dynamic system (1.6) can be discretized as follows:

$$\begin{aligned} \mathbf{y}_{k+1} = & \mathbf{y}_k + g\mathbf{u}_k, \quad k \in [kg, (k+1)g) \\ \mathbf{z}_{k+1} = & \Theta_{k+1}\mathbf{y}_{k+1} - \boldsymbol{\eta}_{k+1} = \boldsymbol{\varepsilon}_{k+1}. \end{aligned} \quad (1.14)$$

The realization of ESEND model (1.13) for computing nonstationary QP problem (1.1) aided with (1.14) regarded as a control system is shown in Fig. 1.2. From the figure, it can be found that ESEND model (1.13) can be deemed as a generalized

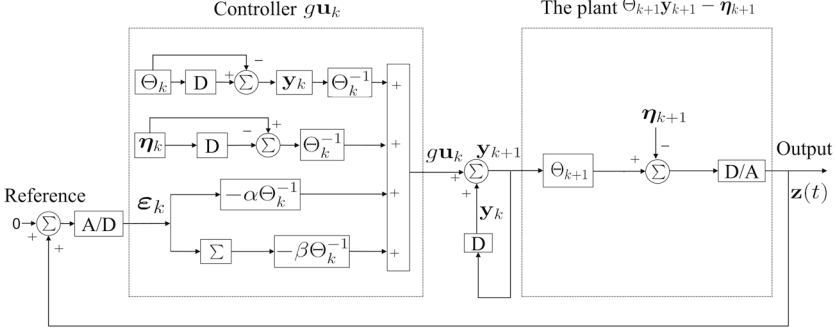


Fig. 1.2 Realizations of ESEND model (1.13) for computing nonstationary QP problem (1.1) regarded as a feedback control system, of which the plant denotes the problem to be solved

discrete proportional-integral-derivative (PID) controller with the control inputs for the derivative, the proportional, and the integral parts being the Euler forward difference of $\dot{\mathbf{y}}_k$, ϵ_{k+1} and $\sum_{j=0}^k (\Theta_k \mathbf{y}_k - \eta_k)$, respectively.

Moreover, five-step forward difference formula (1.11) and five-step backward difference formula are utilized to discretize (1.9), which yields

$$\begin{aligned} \mathbf{y}_{k+1} = & \frac{5}{24} \mathbf{y}_k + \Theta_k^{-1} \left(\frac{1}{24} \Theta_k \check{\mathbf{y}}_k + \frac{1}{30} \check{\eta}_k - \frac{1}{30} \check{\Theta}_k \mathbf{y}_k - \right. \\ & \left. 2\alpha (\Theta_k \mathbf{y}_k - \eta_k) - 2\beta \sum_{j=0}^k (\Theta_k \mathbf{y}_k - \eta_k) \right), \end{aligned} \quad (1.15)$$

with

$$\begin{aligned} \check{\eta}_k &= 137\eta_k - 300\eta_{k-1} + 300\eta_{k-2} - 200\eta_{k-3} + \\ & 75\eta_{k-4} - 12\eta_{k-5}, \\ \check{\Theta}_k &= 137\Theta_k - 300\Theta_{k-1} + 300\Theta_{k-2} - 200\Theta_{k-3} + \\ & 75\Theta_{k-4} - 12\Theta_{k-5}, \\ \check{\mathbf{y}}_k &= 12\mathbf{y}_{k-1} + 6\mathbf{y}_{k-2} + 4\mathbf{y}_{k-3} - 3\mathbf{y}_{k-4}. \end{aligned}$$

The model is just referred to as the five-step neural dynamics (FSND) model in this chapter. It can be seen from this model that the solution at each moment is related to the solutions at previous several moments, that is, each solution process contains multiple interaction subprocesses, which infers that the model (1.15) can be regarded as a stiff system. Afterwards, a corresponding remark on interpreting the stiffness of model (1.15) is given.

Remark 1 A stiff system is a differential equation that includes some terms leading to rapid variation in the solution. Moreover, any continuous-time control system may encounter such a stiffness problem when being discretized. Usually, a small enough step size could conquer the instability of numerical methods resulting from the stiffness of the system.

A discrete-time model is presented in [24] based on (1.10) as well as (1.7), which is rewritten here for comparison:

$$\mathbf{y}_{k+1} = \mathbf{y}_k + \Theta_k^{-1} \left(-g \dot{\Theta}_k \mathbf{y}_k + g \dot{\boldsymbol{\eta}}_k - \alpha (\Theta_k \mathbf{y}_k - \boldsymbol{\eta}_k) \right). \quad (1.16)$$

Applying the Euler forward difference formula (1.10) to the gradient-based neural network (1.4) generates

$$\mathbf{y}_{k+1} = \mathbf{y}_k - \hbar \Theta_k^T (\Theta_k \mathbf{y}_k - \boldsymbol{\eta}_k), \quad (1.17)$$

with $\hbar = \widehat{\gamma}g$ being bounded to $(0, 2/\Gamma_{\max}(\Theta^T \Theta))$, where $\Gamma_{\max}(\Theta^T \Theta)$ is the largest eigenvalue of $\Theta^T \Theta$ [25]. The above equation is named as traditional gradient-based neural network (TGBNN) introduced for comparison purpose.

1.2.3 Compact Reformulation

In this part, to lay a basis for further analyses, the constructed ESEND model (1.13) is reformulated into a compact form with the aid of the following theorem.

Theorem 1.1 *The designed ESEND model (1.13) can be reformulated compactly as*

$$(1 + \alpha + \beta) \boldsymbol{\varepsilon}_k + (\beta - 1) \boldsymbol{\varepsilon}_{k-1} + \beta \sum_{i=0}^{k-2} \boldsymbol{\varepsilon}_i + \mathbf{O}(g^2) = \mathbf{0},$$

where $\mathbf{O}(g^2)$ signifies a vector with each element being $O(g^2)$.

Proof The ESEND model (1.13) can be reformulated as

$$\begin{aligned} \Theta_k (\mathbf{y}_{k+1} - \mathbf{y}_k) &= \boldsymbol{\eta}_k - \boldsymbol{\eta}_{k-1} - (\Theta_k - \Theta_{k-1}) \mathbf{y}_k \\ &\quad - \alpha \boldsymbol{\varepsilon}_k - \beta \sum_{i=0}^k \boldsymbol{\varepsilon}_i. \end{aligned}$$

Simplifying the above equation contributes

$$\Theta_k \mathbf{y}_{k+1} = \boldsymbol{\eta}_k - \boldsymbol{\eta}_{k-1} + \Theta_{k-1} \mathbf{y}_k - \alpha \boldsymbol{\varepsilon}_k - \beta \sum_{i=0}^k \boldsymbol{\varepsilon}_i. \quad (1.18)$$

The following two equations can be derived by leveraging the Taylor expansion:

$$\mathbf{y}_{k+1} = \mathbf{y}_k + g\dot{\mathbf{y}}_k + \frac{g^2\ddot{\mathbf{y}}_k}{2} + \cdots = \mathbf{y}_k + g\dot{\mathbf{y}}_k + \mathbf{O}(g^2), \quad (1.19)$$

and

$$\mathbf{y}_{k-1} = \mathbf{y}_k - g\dot{\mathbf{y}}_k + \frac{g^2\ddot{\mathbf{y}}_k}{2} - \cdots = \mathbf{y}_k - g\dot{\mathbf{y}}_k + \mathbf{O}(g^2),$$

of which the latter one can be formulated as

$$\mathbf{y}_k = \mathbf{y}_{k-1} + g\dot{\mathbf{y}}_k + \mathbf{O}(g^2). \quad (1.20)$$

The following equation holds by substituting (1.19) and (1.20) into (1.18):

$$\begin{aligned} \Theta_k(\mathbf{y}_k + g\dot{\mathbf{y}}_k + \mathbf{O}(g^2)) &= \boldsymbol{\eta}_k - \boldsymbol{\eta}_{k-1} \\ &+ \Theta_{k-1}(\mathbf{y}_{k-1} + g\dot{\mathbf{y}}_k + \mathbf{O}(g^2)) - \alpha\boldsymbol{\varepsilon}_k - \beta \sum_{i=0}^k \boldsymbol{\varepsilon}_i. \end{aligned} \quad (1.21)$$

Simplifying the above equation leads to

$$\begin{aligned} \Theta_k\mathbf{y}_k - \boldsymbol{\eta}_k + g\Theta_k\dot{\mathbf{y}}_k + \mathbf{O}(g^2) &= \Theta_{k-1}\mathbf{y}_{k-1} - \boldsymbol{\eta}_{k-1} \\ &+ g\Theta_{k-1}\dot{\mathbf{y}}_k - \alpha\boldsymbol{\varepsilon}_k - \beta \sum_{i=0}^k \boldsymbol{\varepsilon}_i, \end{aligned} \quad (1.22)$$

which can be further rewritten as

$$\boldsymbol{\varepsilon}_k + g\Theta_k\dot{\mathbf{y}}_k + \mathbf{O}(g^2) = \boldsymbol{\varepsilon}_{k-1} + g\Theta_{k-1}\dot{\mathbf{y}}_k - \alpha\boldsymbol{\varepsilon}_k - \beta \sum_{i=0}^k \boldsymbol{\varepsilon}_i. \quad (1.23)$$

Exploiting Taylor expansion to approximate Θ_k generates

$$\Theta_k = \Theta_{k-1} + g\dot{\Theta}_{k-1} + \mathcal{O}(g^2),$$

where $\mathcal{O}(g^2)$ indicates a matrix-valued truncation error term with each element being $O(g^2)$. Then, based on Eq. (1.23) it can be formulated as

$$\begin{aligned} \boldsymbol{\varepsilon}_k + g(\Theta_{k-1} + g\dot{\Theta}_{k-1} + \mathcal{O}(g^2))\dot{\mathbf{y}}_k + \mathbf{O}(g^2) &= \boldsymbol{\varepsilon}_{k-1} \\ &+ g\Theta_{k-1}\dot{\mathbf{y}}_k - \alpha\boldsymbol{\varepsilon}_k - \beta \sum_{i=0}^k \boldsymbol{\varepsilon}_i, \end{aligned}$$

which is further rewritten as

$$(1 + \alpha + \beta)\mathbf{\varepsilon}_k + (\beta - 1)\mathbf{\varepsilon}_{k-1} + \beta \sum_{i=0}^{k-2} \mathbf{\varepsilon}_i + \mathbf{O}(g^2) = 0. \quad (1.24)$$

The proof is thus completed.

It can be generalized from Theorem 1.1 that Eq. (1.24) is an equivalent formation to the constructed ESEND model (1.13), which can be classified into the category of the linear system in controlology and cybernetics. In what follows, an unknown additive noise $\rho(t)$ is injected into the presented ESEND model (1.13) for laying the foundation to investigate its inherent noise-tolerance capacity:

$$\begin{aligned} \mathbf{y}_{k+1} = & \mathbf{y}_k + \Theta_k^{-1}((\Theta_{k-1} - \Theta_k)\mathbf{y}_k + \boldsymbol{\eta}_k - \boldsymbol{\eta}_{k-1} \\ & - \alpha(\Theta_k \mathbf{y}_k - \boldsymbol{\eta}_k) - \beta \sum_{j=0}^k (\Theta_k \mathbf{y}_k - \boldsymbol{\eta}_k) + \rho(t)). \end{aligned} \quad (1.25)$$

Theorem 1.2 *Discrete-time model (1.16) can be reformulated compactly as $\mathbf{\varepsilon}_{k+1} = (1 - \alpha)\mathbf{\varepsilon}_k + \mathbf{O}(g^2)$.*

Proof It can be generalized from the proof on Theorem 1.1 and thus omitted.

1.3 Theoretical Verifications

Regarding the given ESEND model (1.13), the corresponding theoretical analyses are conducted to draw the following theoretical conclusions.

Theorem 1.3 *The residual error $\lim_{k \rightarrow \infty} \|\mathbf{\varepsilon}_k\|_2$ of the constructed ESEND model (1.13) for computing nonstationary QP problem (1.1) approaches to $O(g^2)$ globally.*

Proof The following expression can be derived from (1.24):

$$(1 + \alpha)\mathbf{\varepsilon}_{k+1} - \mathbf{\varepsilon}_k + \beta \sum_{i=0}^{k+1} \mathbf{\varepsilon}_i + \mathbf{O}(g^2) = 0. \quad (1.26)$$

Let $\mathbf{\varepsilon}_k^j$ denote the j th entry of $\mathbf{\varepsilon}_k$ with $j \in \{1, \dots, n+m\}$. Then, letting the j th row of (1.24) subtract the j th one of (1.26) produces

$$(1 + \alpha + \beta)\mathbf{\varepsilon}_{k+1}^j = (2 + \alpha)\mathbf{\varepsilon}_k^j - \mathbf{\varepsilon}_{k-1}^j + O(g^2). \quad (1.27)$$

Define a new variable ξ_{k+1}^j , which is constituted of $\mathbf{e}_{k+1}^j$ and $\mathbf{e}_k^j$, i.e., $\xi_{k+1}^j = [\mathbf{e}_{k+1}^j, \mathbf{e}_k^j]^T$. Besides, a new matrix Ω is constructed as

$$\Omega = [(2 + \alpha)/(1 + \alpha + \beta) \quad -1/(1 + \alpha + \beta); \quad 1 \quad 0].$$

One can readily deduce that, with ω_1 and ω_2 representing the eigenvalues of Ω , $\omega_1 = (\alpha - \sqrt{\alpha - 4\beta + 2})/(2\alpha + 2\beta + 2)$ and $\omega_2 = (\alpha + \sqrt{\alpha - 4\beta + 2})/(2\alpha + 2\beta + 2)$. Consequently, a conclusion is drawn that $|\text{Re}(\omega_1)| < 1$ and $|\text{Re}(\omega_2)| < 1$, with $\text{Re}(\omega_1)$ (or $\text{Re}(\omega_2)$) the real part of ω_1 (or that of ω_2). Then, Eq. (1.27) can be rewritten in the form of state-space equation as $\xi_{k+1}^j = \Omega \xi_k^j + O(g^2)$, from which, based on the triangle inequality, one can further generalize

$$\begin{aligned} \|\xi_{k+1}^j\|_2 &\leq \|\Omega \xi_k^j\|_2 + \|O(g^2)\|_2 \\ &= \|\Omega \xi_k^j\|_2 + O(g^2) \\ &\leq \|\Omega\|_2 \|\xi_k^j\|_2 + O(g^2) \\ &\leq \|\Omega\|_2 \|\Omega \xi_{k-1}^j\|_2 + \|\Omega\|_2 O(g^2) + O(g^2) \\ &= \|\Omega\|_2 \|\Omega \xi_{k-1}^j\|_2 + O(g^2) \\ &\leq \|\Omega\|_2^2 \|\xi_{k-1}^j\|_2 + O(g^2) \\ &\vdots \\ &\leq \|\Omega\|_2^k \|\xi_1^j\|_2 + O(g^2). \end{aligned}$$

Thus, it has $\lim_{k \rightarrow \infty} \|\Omega\|_2^k = 0$ for $|\text{Re}(\omega_1)| < 1$ and $|\text{Re}(\omega_2)| < 1$ and further has

$$\lim_{k \rightarrow \infty} \|\xi_{k+1}^j\|_2 \leq \lim_{k \rightarrow \infty} \|\Omega\|_2^k \|\xi_1^j\|_2 + O(g^2) = O(g^2).$$

Recalling the definition of ξ_{k+1}^j , i.e., $\xi_{k+1}^j = [\mathbf{e}_{k+1}^j, \mathbf{e}_k^j]^T$, $\lim_{k \rightarrow \infty} \|\mathbf{e}_k\|_2 = O(g^2)$ is obtained. One can readily get the conclusion that the residual error of the presented ESEND model (1.13) for computing nonstationary QP problem (1.1) approaches to $O(g^2)$ globally. The proof is thus completed.

Corollary *The first n elements of neural state $\mathbf{y}_k$ of the designed ESEND model (1.13) globally approach to the theoretical ones to (1.1) with upper bound of the error being $O(g^2)$.*

Theorem 1.4 *Given that $0 < \alpha < 2$, the residual error $\lim_{k \rightarrow \infty} \|\mathbf{e}_k\|_2$ of discrete-time model (1.16) for computing nonstationary QP problem (1.1) approaches to $O(g^2)$ globally.*

Proof The proof can be divided into the following three subcases: (i) $\alpha = 1$, (ii) $\epsilon_0 = 0$, and (iii) the rest. For the above three subcases, the following theoretical results are offered:

- (i) As to the subcase of $\alpha = 1$, $\epsilon_{k+1} = (1 - \alpha)\epsilon_k + \mathbf{O}(g^2) = \mathbf{O}(g^2)$.
- (ii) As to the subcase of $\epsilon_0 = 0$,

$$\begin{aligned}
 \epsilon_{k+1}^j &= (1 - \alpha)\epsilon_k^j + O(g^2) \\
 &= (1 - \alpha)((1 - \alpha)\epsilon_{k-1}^j + O(g^2)) + O(g^2) \\
 &= (1 - \alpha)^2\epsilon_{k-1}^j + (1 - \alpha)O(g^2) + O(g^2) \\
 &= (1 - \alpha)^3\epsilon_{k-2}^j + (1 - \alpha)^2O(g^2) + (1 - \alpha)O(g^2) \\
 &\quad + O(g^2) \\
 &\quad \vdots \\
 &= (1 - \alpha)^{k+1}\epsilon_0^j + (1 - \alpha)^kO(g^2) + \cdots \\
 &\quad + (1 - \alpha)^1O(g^2) + O(g^2) \\
 &= (1 - \alpha)^kO(g^2) + \cdots + (1 - \alpha)^1O(g^2) + O(g^2) \\
 &= \frac{1 - (1 - \alpha)^{k+1}}{\alpha}O(g^2) \\
 &= O(g^2).
 \end{aligned}$$

- (iii) As to the subcase of $\epsilon_0 \neq 0$ and $\alpha \neq 1$,

$$\begin{aligned}
 \epsilon_{k+1}^j &= (1 - \alpha)\epsilon_k^j + O(g^2) \\
 &= (1 - \alpha)((1 - \alpha)\epsilon_{k-1}^j + O(g^2)) + O(g^2) \\
 &\quad \vdots \\
 &= (1 - \alpha)^{k+1}\epsilon_0^j + (1 - \alpha)^kO(g^2) + \cdots \\
 &\quad + (1 - \alpha)^1O(g^2) + O(g^2) \\
 &= (1 - \alpha)^{k+1}\epsilon_0^j + \frac{1 - (1 - \alpha)^{k+1}}{\alpha}O(g^2) \\
 &= (1 - \alpha)^{k+1}\epsilon_0^j + O(g^2).
 \end{aligned}$$

In view of $\lim_{k \rightarrow \infty} (1 - \alpha)^{k+1} = 0$ for $\alpha \in (0, 2)$, one obtains

$$\lim_{k \rightarrow \infty} \epsilon_{k+1}^j = \lim_{k \rightarrow \infty} (1 - \alpha)^{k+1}\epsilon_0^j + O(g^2) = O(g^2).$$

One can readily get the conclusion that the residual error of discrete-time model (1.16) for computing the nonstationary QP problem (1.1) approaches to $O(g^2)$ globally. The proof is thus completed.

It is worth noting that, under the hypothesis of short-time invariance, various discrete-time numerical algorithms or continuous-time models are leveraged to solve QP problems online, e.g., Newton iteration, conjugate-gradient method, and their variants. The following theorem and its proof are presented to investigate their performance under a general framework. Hereafter, any computing approach exploiting the hypothesis of short-time invariance is termed as a proportional-type method.

Theorem 1.5 *Given that the proportional-type method converges to the theoretical solution to a stationary QP problem within g , if a proportional-type method is leveraged to compute nonstationary QP problem (1.1), its residual error $\lim_{k \rightarrow \infty} \|\mathbf{e}_k\|_2$ approaches $O(g)$ globally.*

Proof According to the definition of time derivative, one can derive that $\lim_{g \rightarrow 0} \Delta \mathbf{y}_k^j / g = d\mathbf{y}_k^j / dt = p_k^j$ and $\Delta \mathbf{y}_k^j \approx p_k^j g$, where $\mathbf{y}_k^j$ refers to the j th entry of $\mathbf{y}_k$; p_k^j being a constant value denotes the time derivative of $\mathbf{y}^j$ at time instant $t = kg$. Consequently, $\Delta \mathbf{y}_k^j$ is of the $O(g)$ pattern. Put another way, $\Delta \mathbf{y}_k^j = O(g)$ and $\Delta \mathbf{y}_k = \mathbf{O}(g)$. In addition, during time interval $[kg, (k+1)g)$, the proportional-type method approaches the theoretical solution denoted by $\mathbf{y}_k^*$ at $t = kg$ and $\mathbf{y}_{k+1}^* = \mathbf{y}_k^* + \Delta \mathbf{y}_k$. Therefore, at $t = (k+1)g$, the difference between the computed solution via the proportional-type method and the theoretical one is $\Delta \mathbf{y}_k$, which can be formulated as $\mathbf{y}_{k+1}^* = \mathbf{y}_k^* + \mathbf{O}(g) = \mathbf{y}_{k+1} + \mathbf{O}(g)$. One has

$$\begin{aligned}
 \|\mathbf{e}_{k+1}\|_2 &= \|\Theta_{k+1}\mathbf{y}_{k+1} - \boldsymbol{\eta}_{k+1}\|_2 \\
 &= \|\Theta_{k+1}\mathbf{y}_{k+1}^* - \boldsymbol{\eta}_{k+1} + \Theta_{k+1}\mathbf{O}(g)\|_2 \\
 &= \|\mathbf{0} + \Theta_{k+1}\mathbf{O}(g)\|_2 \\
 &= \|\Theta_{k+1}\mathbf{O}(g)\|_2 \\
 &= \|\mathbf{O}(g)\|_2 \\
 &= O(g).
 \end{aligned}$$

The proof is thus completed.

About the constructed ESEND model (1.13), theoretical results on its robustness are given as below.

Theorem 1.6 *Perturbed by an unknown constant-biased noise $\rho(t) = v$, the residual error $\lim_{k \rightarrow \infty} \|\mathbf{e}_k\|_2$ of the presented ESEND model (1.13) for computing nonstationary QP problem (1.1) approaches $O(g^2)$ globally.*

Proof Being the compact form of the presented ESEND model (1.13), Eq. (1.24) can be rewritten as the combination of a linear system $\mathcal{L} = (1 + \alpha + \beta)\mathbf{e}_k +$

$(\beta - 1)\mathbf{e}_{k-1} + \beta \sum_{i=0}^{k-2} \mathbf{e}_i$ and the residual error $\mathbf{O}(g^2)$. For the linear system $\mathcal{L}$, the principle of superposition is applied and thus the robustness of the constructed ESEND model (1.13) can be analyzed by separately handling the subsystem perturbed by constant-biased noise $\rho(t) = v$ and that perturbed by the residual error $\mathbf{O}(g^2)$. In addition, as proven in the previous theorem, the residual error of the given ESEND model (1.13) for computing nonstationary QP problem (1.1) approaches $\mathbf{O}(g^2)$ globally. Now, consider the following subsystem:

$$(1 + \alpha + \beta)\mathbf{e}_k + (\beta - 1)\mathbf{e}_{k-1} + \beta \sum_{i=0}^{k-2} \mathbf{e}_i + v = 0,$$

which is equivalent to the following one:

$$(1 + \alpha + \beta)\mathbf{e}_{k+1} - \mathbf{e}_k + \beta \sum_{i=0}^k \mathbf{e}_i + v = 0. \quad (1.28)$$

The j th subsystem of (1.28) is

$$(1 + \alpha + \beta)\mathbf{e}_{k+1}^j - \mathbf{e}_k^j + \beta \sum_{i=0}^k \mathbf{e}_i^j + v^j = 0. \quad (1.29)$$

Leveraging Z-transform to the above equation leads to

$$(1 + \alpha + \beta)(z\mathbf{e}^j(z) - z\mathbf{e}^j(0)) - \mathbf{e}^j(z) + \beta \frac{\mathbf{e}^j(z)}{1 - z^{-1}} - \frac{zv^j}{z - 1} = 0, \quad (1.30)$$

with $\mathbf{e}^j(0)$ the initial value of $\mathbf{e}^j(z)$. Reformulating Eq. (1.30) premises

$$\begin{aligned} & ((1 + \alpha + \beta)z(z - 1) - (z - 1) + z\beta)\mathbf{e}^j(z) \\ & = (1 + \alpha + \beta)z(z - 1)\mathbf{e}^j(0) + zv^j \end{aligned}$$

and further elicits

$$\mathbf{e}^j(z) = \frac{(1 + \alpha + \beta)z(z - 1)\mathbf{e}^j(0) + zv^j}{(1 + \alpha + \beta)z(z - 1) - (z - 1) + z\beta}. \quad (1.31)$$

The poles of the above equation are $z_{1,2} = (\alpha \pm \sqrt{\alpha^2 - 4\beta + 2}) / (2\alpha + 2\beta + 2)$, which are within the unit circle, thereby showing the stability of Eq. (1.29). Leveraging the final value theorem to Eq. (1.31) produces

$$\begin{aligned}
\lim_{k \rightarrow \infty} \mathbf{e}_k^j &= \lim_{z \rightarrow 1} (z - 1) \mathbf{e}^j(z) \\
&= \lim_{z \rightarrow 1} \frac{(z - 1)((1 + \alpha + \beta)z(z - 1) \mathbf{e}^j(0) + z v^j)}{(1 + \alpha + \beta)z(z - 1) - (z - 1) + z\beta}, \\
&= 0.
\end{aligned}$$

By considering the definition of $\mathbf{e}$ as well as the character of linear system $\mathcal{L}$, it comes to the conclusion that, perturbed by an unknown constant-biased noise $\rho(t) = v$, the residual error of the designed ESEND model (1.13) for computing nonstationary QP problem (1.1) approaches $O(g^2)$ globally. The proof is thus completed.

About the robustness of discrete-time model (1.16), the following theoretical results are supplied.

Theorem 1.7 *Perturbed by an unknown constant-biased noise $\rho(t) = v$, the residual error $\lim_{k \rightarrow \infty} \|\mathbf{e}_k\|_2$ of discrete-time model (1.16) for computing nonstationary QP problem (1.1) approaches $\|v\|_2/\alpha + O(g^2)$ globally.*

Proof By following similar procedures as in the proof on Theorem 1.6, it is required to consider the following subsystem:

$$\mathbf{e}_{k+1} - (1 - \alpha)\mathbf{e}_k - v = 0.$$

Employing Z-transform to the j th subsystem of the above equation leads to

$$z \mathbf{e}^j(z) - z \mathbf{e}^j(0) - (1 - \alpha) \mathbf{e}^j(z) - \frac{z v^j}{z - 1} = 0.$$

Manipulating the above equation yields

$$\mathbf{e}^j(z) = \frac{z(z - 1) \mathbf{e}^j(0) + z v^j}{(z - 1)(z - 1 + \alpha)}, \quad (1.32)$$

with the poles being $z_1 = 1$, and $z_2 = 1 - \alpha$. Evidently, there is only one pole on the unit circle with the other one located within it. Using the final value theorem to (1.32) generates

$$\begin{aligned}
\lim_{k \rightarrow \infty} \mathbf{e}_k^j &= \lim_{z \rightarrow 1} (z - 1) \mathbf{e}^j(z) \\
&= \lim_{z \rightarrow 1} (z - 1) \frac{z(z - 1) \mathbf{e}^j(0) + z v^j}{(z - 1)(z - 1 + \alpha)}, \\
&= v^j / \alpha.
\end{aligned}$$

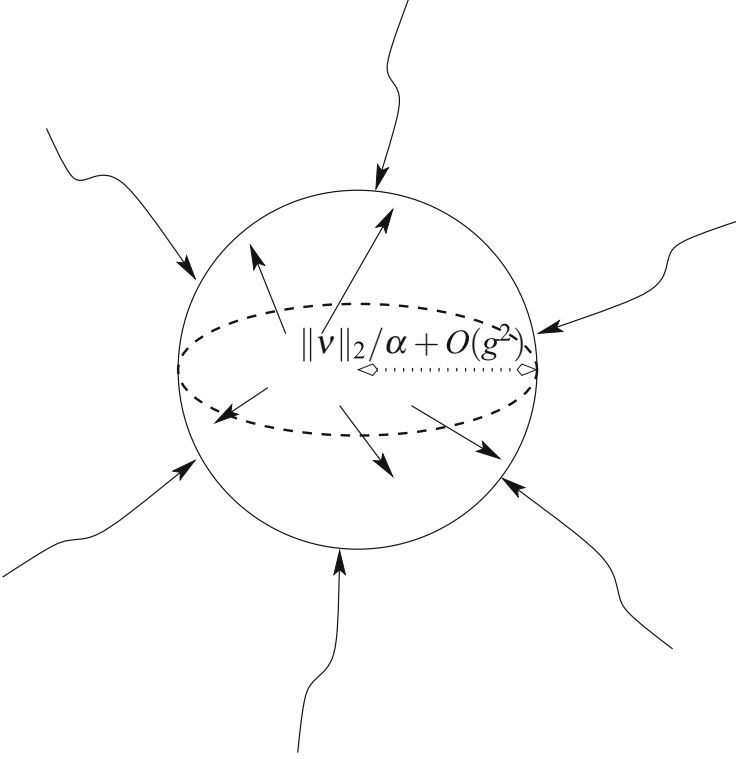


Fig. 1.3 Perturbed by an unknown constant-biased noise $\rho(t) = v$, the residual error $\lim_{k \rightarrow \infty} \|\epsilon(t)\|_2$ of discrete-time model (1.16) for computing nonstationary QP problem (1.1) approaches globally to the surface of a ball with radius being $\|v\|_2/\alpha + O(g^2)$

Similarly to the proof in Theorem 1.6, a conclusion can be drawn that, perturbed by an unknown constant-biased noise $\rho(t) = v$, the residual error of discrete-time model (1.16) for computing nonstationary QP problem (1.1) approaches $\|v\|_2/\alpha + O(g^2)$ globally. Additionally, the convergence performance of discrete-time model (1.16) is vividly expressed in Fig. 1.3 with errors in any scope approaching the spherical surface of radius $\|v\|_2/\alpha + O(g^2)$. The proof is thus completed.

In addition, regarding the performance of the constructed ESEND model (1.13) perturbed by the bounded random noise, the following theorem is provided.

Theorem 1.8 *Perturbed by the bounded random noise $\rho(t)$, the residual error $\lim_{k \rightarrow \infty} \|\epsilon_k\|_2$ of the given ESEND model (1.13) for computing nonstationary QP problem (1.1) approaches to $2(n+m) \sup_{1 \leq i \leq k, 1 \leq j \leq (n+m)} |\rho_i^j| / (1 - \|\Omega\|_2) + O(g^2)$ globally.*

Proof Similar to the proof in Theorem 1.6, the analyses can be conducted by separately considering two subsystems relating to $\mathbf{O}(g^2)$ and the bounded random noise $\rho(t)$. Based on the derivation procedures of Eq. (1.27), the following equation can be readily deduced:

$$(1 + \alpha + \beta)\mathbf{e}_{k+1}^j = (2 + \alpha)\mathbf{e}_k^j - \mathbf{e}_{k-1}^j + \rho_k^j - \rho_{k-1}^j. \quad (1.33)$$

By defining $\mu_k^j = [\rho_k^j - \rho_{k-1}^j, \mathbf{0}]^T$, Eq. (1.33) can be rearranged into a state-space equation in control terminology:

$$\xi_{k+1}^j = \Omega \xi_k^j + \mu_k^j.$$

Accordingly,

$$\begin{aligned} \|\xi_{k+1}^j\|_2 &\leq \|\Omega \xi_k^j\|_2 + \|\mu_k^j\|_2 \\ &\leq \|\Omega\|_2 \|\xi_k^j\|_2 + \|\mu_k^j\|_2 \\ &\leq \|\Omega\|_2 \|\Omega \xi_{k-1}^j\|_2 + \|\Omega\|_2 \|\mu_{k-1}^j\|_2 + \|\mu_k^j\|_2 \\ &\leq \|\Omega\|_2^2 \|\xi_{k-1}^j\|_2 + \|\Omega\|_2 \|\mu_{k-1}^j\|_2 + \|\mu_k^j\|_2 \\ &\quad \vdots \\ &\leq \|\Omega\|_2^k \|\xi_1^j\|_2 + \|\Omega\|_2^{k-1} \|\mu_1^j\|_2 + \cdots + \|\mu_k^j\|_2 \\ &< \|\Omega\|_2^k \|\xi_1^j\|_2 + \max_{1 \leq i \leq k} \|\mu_i^j\|_2 / (1 - \|\Omega\|_2) \\ &< \|\Omega\|_2^k \|\xi_1^j\|_2 + 2 \max_{1 \leq i \leq k} |\mu_i^j| / (1 - \|\Omega\|_2). \end{aligned}$$

Then, $\lim_{k \rightarrow \infty} \|\xi_{k+1}^j\|_2 < 2 \max_{1 \leq i \leq k} |\mu_i^j| / (1 - \|\Omega\|_2)$ owing to $\lim_{k \rightarrow \infty} \|\Omega\|_2^k = 0$. Considering the definition of ξ , one can readily get the conclusion that

$$\lim_{k \rightarrow \infty} \|\mathbf{e}_k\|_2 < 2\iota \sup_{\substack{1 \leq i \leq k \\ 1 \leq j \leq \iota}} |\rho_i^j| / (1 - \|\Omega\|_2) + O(g^2),$$

with $\iota = m + n$. The proof is thus completed.

Up to now, eight theorems and the related proofs have been provided, which convert the discrete-time neural dynamics into a linear system plus a residual term and then present their convergence performance and robustness analyses. To further explore the mechanism leading to different performances in these computing methods, the ensuing comparisons are offered.

- (1) In terms of the formula expressions, computing methods can be classified into two categories: continuous-time approaches and discrete-time approaches. As stated in Sect. 1.2.2, any computing method depicted in continuous-time form for solving nonstationary problems implemented on a digital computer with future information exploited can be described as a noncausal system. Consequently, a suitable numerical differentiation formula is the key to construct a computing method depicted in discrete time complied with the law of causality. In addition, any discrete-time computing method for solving nonstationary problems is a time-delay system and thus the delay error can only be decreased but not be theoretically eliminated.
- (2) In terms of the computing patterns, computing methods can be classified into two categories: the recursive methods and direct methods. The latter often achieve the theoretical solution in manners of error-free and finite steps, which are not able to solve complex problems effectively. On the contrary, the recursive methods recursively estimate a solution based on the error-feedback information generated by the previous data for achieving a given accuracy. From the control perspective, Newton iteration (1.3), the traditional gradient-based neural network (1.17), and other proportional-type computing methods leverage error-feedback information to find an appropriate search direction. It is a common view in the control society that the lagging error is unavoidable for a dynamic system aided with a proportional controller. It is difficult to design a computing method for handling nonstationary problems from the computing framework, whereas many techniques for handling such knotty problems are available in control fields. For example, to improve the computing performance in solving nonstationary problems, an intuitive method is to exploit the time-derivation information.

Remark 2 It is of note that the gradient-based neural network for solving dynamic problems is of large lagging errors due to the absence of the velocity compensation of varying parameters. Besides, this neural network is sensitive to perturbations. Therefore, considering the analyses on the systematic framework in control terminology, the time-derivation information of time-dependent parameters and the error accumulation term are absorbed by improving the performance of TGBNN model (1.17). In this sense, this proportional-type computing method is evolved into the proportional integral differential one for the continuous-time one or proportional summation difference one for the discrete-time one, which is reformulated as

$$\begin{aligned} \Theta(t)\dot{\mathbf{y}}(t) = & -\dot{\Theta}(t)\mathbf{y}(t) + \dot{\boldsymbol{\eta}}(t) - \hat{\gamma}\Theta(t)\Theta^T(t)(\Theta(t)\mathbf{y}(t) - \\ & \boldsymbol{\eta}(t)) - \lambda \int_0^t (\Theta(\delta)\mathbf{y}(\delta) - \boldsymbol{\eta}(\delta))d\delta. \end{aligned}$$

Applying Euler forward difference formula (1.10) to $\dot{\mathbf{y}}(t)$ and Euler backward difference formula to $\dot{\Theta}(t)$ and $\dot{\boldsymbol{\eta}}(t)$ elicits the designed compensatory gradient-based neural dynamics (CGND) model:

$$\begin{aligned} \mathbf{y}_{k+1} = & \mathbf{y}_k + \Theta_k^{-1}((\Theta_{k-1} - \Theta_k)\mathbf{y}_k + \boldsymbol{\eta}_k - \boldsymbol{\eta}_{k-1} \\ & - h\Theta_k\Theta_k^T(\Theta_k\mathbf{y}_k - \boldsymbol{\eta}_k) - \beta \sum_{j=0}^k (\Theta_k\mathbf{y}_j - \boldsymbol{\eta}_j)). \end{aligned} \quad (1.34)$$

Remark 3 It is deserved to note that the constructed ESEND model (1.13) is equipped with high robustness perturbed with kinds of noises, which is mainly attributed to the summation item (i.e., $\sum_{j=0}^k (\Theta_k\mathbf{y}_j - \boldsymbol{\eta}_j)$). Similarly, considering the important role of the integral control played in the controllers, integrating this term to the classical Newton iteration (1.3) to endow it with noise-suppression property contributes

$$\mathbf{y}_{k+1} = \mathbf{y}_k - \Theta_k^{-1}(\Theta_k\mathbf{y}_k - \boldsymbol{\eta}_k + \beta \sum_{j=0}^k (\Theta_k\mathbf{y}_j - \boldsymbol{\eta}_j)), \quad (1.35)$$

which is called as error-accumulation-considered Newton iteration (EANI) model in the following content. Namely, this proportional-type computing method is developed into the proportional-integral one for the continuous-time one or proportional summation one for the discrete-time one.

Therefore, from the viewpoint of control, the given CGND model (1.34) and the designed EANI model (1.35) are classified into proportional summation difference controllers and proportional summation controllers, respectively, which can be deduced by inspecting the construction process of the constructed CGND model (1.34) similar to Fig. 1.2 and that of the given EANI model (1.35) described in Fig. 1.4. Combined with the structure of the designed ESEND model (1.13), which falls into proportional summation difference controllers, these three computing methods constructed are different in the formula formations, derivation approaches, and construction techniques, but share the same control technology in nonstationary problem-solving and noise elimination. Thus, the aforementioned models (1.13), (1.34), and (1.35) are appropriate for the perturbed nonstationary QP.

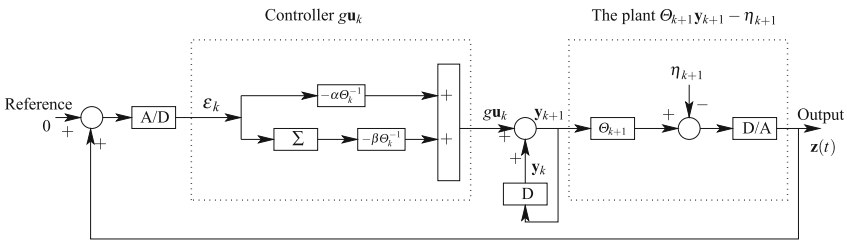


Fig. 1.4 Realizations of EANI model (1.35) for computing nonstationary QP problem (1.1) regarded as a feedback control system, of which the definitions on the controller $g\mathbf{u}_k$ and the plant (the problem to be solved) $\Theta_{k+1}\mathbf{y}_{k+1} - \boldsymbol{\eta}_{k+1}$ are the same as those in Eq. (1.14)

1.4 Illustrative Examples

In this section, an illustrative numerical example and a simulation application in robots are provided and analyzed to substantiate the efficacy, superiority, and physical realizability of the given ESEND model (1.13), the designed CGND model (1.34), and the constructed EANI model (1.35) for solving perturbed nonstationary QP problem (1.1).

1.4.1 Example 1

In this section, specializing the discussion of Sect. 1.2.1, the nonstationary QP problem (1.2) with parameters $n = 2$ and $m = 1$ is considered for illustration and for comparison with the following coefficients:

$$\Theta(t) = \begin{bmatrix} \sin(t) + 10 & \cos(t) & \sin(4t) \\ \cos(t) & \cos(t) + 10 & \cos(4t) \\ \sin(4t) & \cos(4t) & 10 \end{bmatrix}, \quad (1.36)$$

and $\eta(t) = [\sin(6t) \cos(6t) - \cos(2t)]^T$.

In what follows, the adopted parameters are set as $\alpha = 1$; $\beta = 25$; $h_k = 1.9/\Gamma_{\max}(\Theta_k^T \Theta_k)$, randomly generated initial values $\mathbf{y}_0 \in [0, 1]^3$. Thus, numerical experiments are conducted with the corresponding results shown in Figs. 1.5 and 1.6 as well as Tables 1.1 and 1.2. For comparison, the given ESEND model (1.13), the designed CGND model (1.34), the constructed EANI model (1.35), Newton iteration (1.3), discrete-time model (1.16), and TGBNN model (1.17) are employed to solve the same nonstationary QP problem (1.36) from two perspectives: (1) without noises and (2) with noises (constant noises, linear time-dependent noises, random noises). It is clearly shown in Figs. 1.5 and 1.6 as well as Table 1.1 that for (1), all the six models can converge to the theoretical solution $\mathbf{y}_k^*$ among which the given ESEND model (1.13) and the designed CGND model (1.34) manifest the superior convergence performance with the maximal steady-state residual errors (MSSREs) of the order 10^{-4} for $g = 0.01$ s or 10^{-6} for $g = 0.001$ s. Combined with the simulation results for $g = 0.0001$ s listed in Table 1.1, the conclusion can be drawn that firstly, residual errors synthesized by the constructed ESEND model (1.13) and the discrete-time model (1.16) change in an $O(g^2)$ manner, while those of Newton iteration (1.3) and TGBNN model (1.17) change in an $O(g)$ manner, which demonstrates the validity and correctness of Theorems 1.3 and 1.5 and embodies the excellent convergence performance of the given ESEND model (1.13). Secondly, compared with Newton iteration (1.3) and TGBNN model (1.17), the designed CGND model (1.34) and the constructed EANI model (1.35) greatly reduce the residual error and further improve the accuracy of the computing process, which benefits from the perfection of the model structure. In this sense, it accords

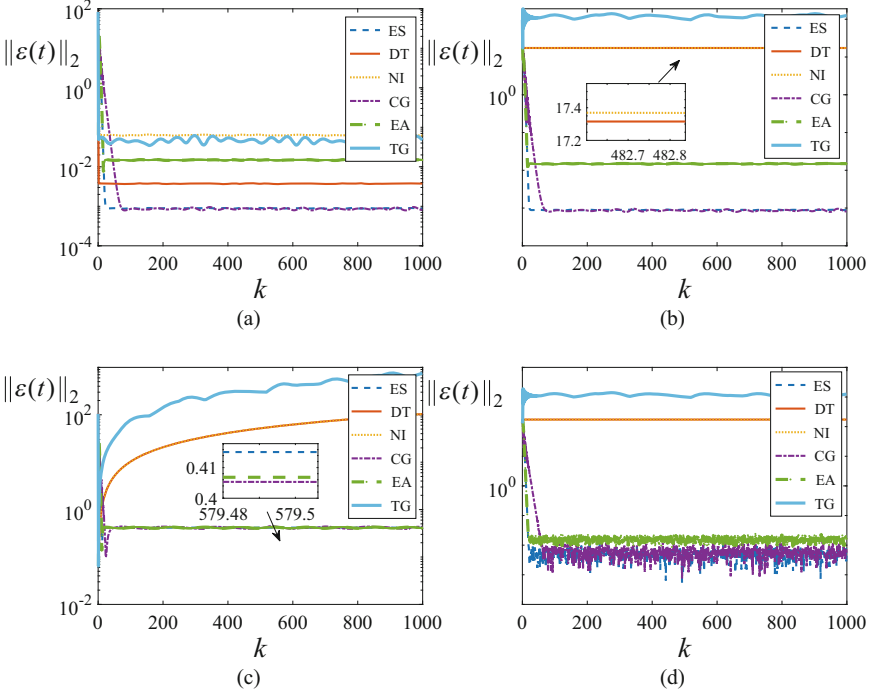


Fig. 1.5 Residual errors synthesized by the given ESEND model (1.13) (represented by ES), the designed CGND model (1.34) (represented by CG), the constructed EANI model (1.35) (represented by EA), Newton iteration (1.3) (represented by NI), discrete-time model (1.16) (represented by DT), and TGBNN model (1.17) (represented by TG) for computing nonstationary QP problem (1.36) with $g = 0.01$ s, $\alpha = 1$, $\beta = 25$, and randomly generated initial values $\mathbf{y}_0 \in [0, 1]^3$. (a) Profiles of MSSRE without noises, (b) profiles of MSSRE with randomly generated constant noises, (c) profiles of MSSRE with linear time-dependent noises $\rho(t) = 6 * (t - 1)$, and (d) profiles of MSSRE with random noises $\rho(t) \in [99.5, 100.5]$

with that the proportional summation difference controller is more stable than the proportional summation controller and the proportional difference controller from the perspective of control.

In addition, for (2), as depicted in (b) through (d) in Figs. 1.5 and 1.6, the given ESEND model (1.13), the designed CGND model (1.34) and the constructed EANI model (1.35) can handle the three kinds of noises all the time and force the residual errors to maintain in tiny values, while the other models are invalid with large errors (or going infinity with time) and fail to solve the nonstationary QP problem (1.36). Particularly, perturbed by randomly generated constant noises, residual errors synthesized by the given ESEND model (1.13), the designed CGND model (1.34), and the constructed EANI model (1.35) all change in an $O(g^2)$ manner as demonstrated in Theorems 1.6 and 1.7. Moreover, it is obtained from the var figures and Table 1.1 that residual errors of the given ESEND model (1.13)

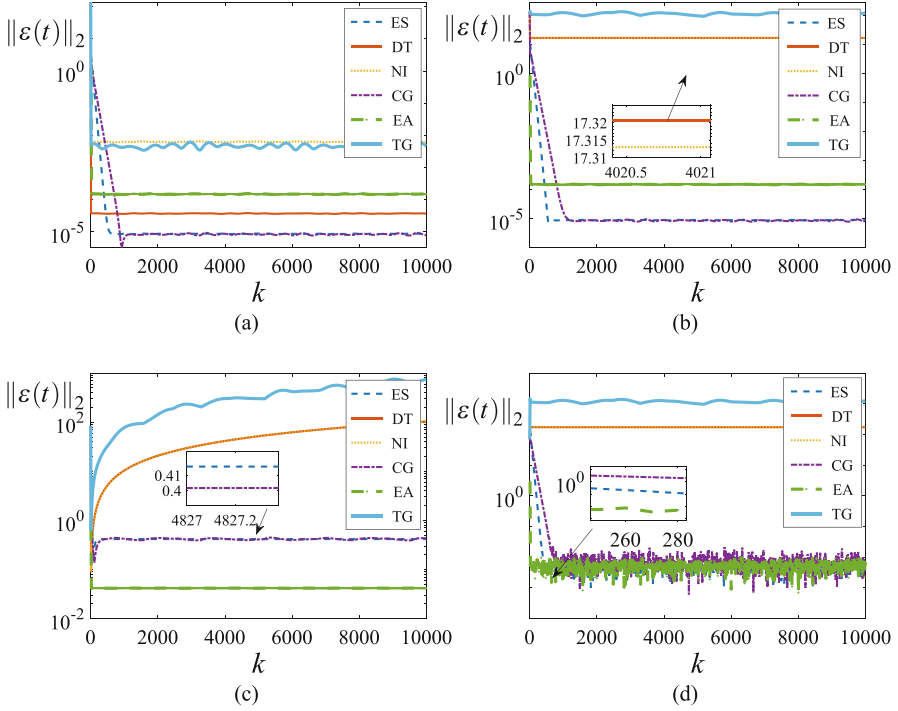


Fig. 1.6 Residual errors synthesized by the given ESEND model (1.13) (represented by ES), the designed CGND model (1.34) (represented by CG), the constructed EANI model (1.35) (represented by EA), Newton iteration (1.3) (represented by NI), discrete-time model (1.16) (represented by DT), and TGBNN model (1.17) (represented by TG) for computing nonstationary QP problem (1.36) with $g = 0.001$ s, $\alpha = 1$, $\beta = 25$, and randomly generated initial values $\mathbf{y}_0 \in [0, 1]^3$. (a) Profiles of MSSRE without noises, (b) profiles of MSSRE with randomly generated constant noises, (c) profiles of MSSRE with linear time-dependent noises $\rho(t) = 6 * (t - 1)$, and (d) profiles of MSSRE with random noises $\rho(t) \in [99.5, 100.5]$

and the designed CGND model (1.34) are almost 10^9 times less than those of Newton iteration (1.3), discrete-time model (1.16), and TGBNN model (1.17) at $g = 0.0001$ s. The analysis on the cases of perturbed by linear time-dependent noises and random noises is similar to that on the above situation and thus omitted here. The simulation experimental results reflect the superiority and effectiveness of the constructed ESEND model (1.13), the given CGND model (1.34), and the designed EANI model (1.35) for computing nonstationary QP problem (1.36) in terms of the steady-state residual error and robustness.

Considering that FSND model (1.15) is a stiff system, it cannot be used to solve nonstationary QP problem (1.36) with infinite residual errors under the same parameter $\alpha = 1$, as shown in Table 1.2. According to Remark 1, it can be seen that when step size drops to $\alpha = 0.2$, FSND model (1.15) can perform the solution task

Table 1.1 Comparisons among different models (the given ESEND model (1.13), the designed CGND model (1.34), the constructed EANI model (1.35), Newton iteration (1.3), discrete-time model (1.16), and TGBNN model (1.17)) for computing nonstationary QP problem (1.36) in terms of MSSRE, where their initial states are generated randomly

g (s)	Model	MSSRE without noises	MSSRE with randomly generated constant noises	MSSRE with linear noises $\rho(t) = 6 * (t - 1)$	MSSRE with random noises $\rho(t) \in [99.5, 100.5]$
0.01	Model (1.3)	6.5700×10^{-2}	1.7417×10^1	Infinity	1.7378×10^2
	Model (1.13)	9.0563×10^{-4}	9.0563×10^{-4}	4.1640×10^{-1}	9.5000×10^{-3}
	Model (1.16)	3.8000×10^{-3}	1.7325×10^1	Infinity	1.7325×10^2
	Model (1.17)	6.1700×10^{-2}	1.4041×10^2	7.6081×10^2	1.4041×10^3
	Model (1.34)	9.5725×10^{-4}	9.5725×10^{-4}	4.3740×10^{-1}	1.4700×10^{-2}
	Model (1.35)	1.5200×10^{-2}	1.5200×10^{-2}	4.2860×10^{-1}	2.2600×10^{-2}
0.001	Model (1.3)	6.6000×10^{-3}	1.7330×10^1	Infinity	1.7328×10^2
	Model (1.13)	8.8909×10^{-6}	8.8909×10^{-6}	4.1570×10^{-1}	8.8000×10^{-3}
	Model (1.16)	3.8332×10^{-5}	1.7321×10^1	Infinity	1.7322×10^2
	Model (1.17)	6.2000×10^{-3}	1.4016×10^2	7.5928×10^2	1.4017×10^3
	Model (1.34)	9.3513×10^{-6}	9.3513×10^{-6}	4.3660×10^{-1}	2.2000×10^{-2}
	Model (1.35)	1.5332×10^{-4}	1.5332×10^{-4}	4.170×10^{-2}	1.0400×10^{-2}
0.0001	Model (1.3)	6.5682×10^{-4}	1.7322×10^1	Infinity	1.7323×10^2
	Model (1.13)	8.8745×10^{-8}	8.8745×10^{-8}	4.1570×10^{-1}	8.5000×10^{-3}
	Model (1.16)	3.8331×10^{-7}	1.7321×10^1	Infinity	1.7322×10^2
	Model (1.17)	6.1762×10^{-4}	1.4013×10^2	7.5913×10^2	1.4015×10^3
	Model (1.34)	9.3300×10^{-8}	9.3300×10^{-8}	4.3650×10^{-1}	2.5600×10^{-2}
	Model (1.35)	1.5332×10^{-6}	1.5332×10^{-6}	4.2000×10^{-3}	1.1400×10^{-2}

effectively and accurately and its residual error changes in an $O(g^4)$ manner, which validates the correctness of the descriptions in Sect. 1.2.2 and Remark 1.

1.4.2 Example 2 (Application in Robots)

In this part, the designed ESEND model (1.13), the constructed CGND model (1.34), and the given EANI model (1.35) are applied to the motion generation of a six-link planar robot manipulator in terms of the kinematic control of redundant robot manipulators in the general framework of optimization and the QP widely involved in the control field [26–30]. Explicitly, the affine system at the joint-velocity level originating from the forward kinematics equation of the UR5 robot is specified as

$$J(\theta(t))\dot{\theta}(t) = \dot{\varpi}(t), \quad (1.37)$$

Table 1.2 MSSREs of FSND model (1.15) for computing nonstationary QP problem (1.36) with different step size α

	g (s)	Step size α				
		$\alpha = 1$	$\alpha = 0.6$	$\alpha = 0.2$	$\alpha = 0.1$	$\alpha = 0.05$
MSSRE without noises	0.01	NA	NA	1.1067×10^{-5}	1.2559×10^{-5}	1.3290×10^{-5}
	0.001	NA	NA	1.0013×10^{-9}	1.0907×10^{-9}	1.1181×10^{-9}
	0.0001	NA	NA	4.4012×10^{-13}	3.5764×10^{-13}	3.2637×10^{-13}
MSSRE with randomly generated constant noises	0.01	NA	NA	1.1067×10^{-5}	1.2559×10^{-5}	1.3290×10^{-5}
	0.001	NA	NA	1.0014×10^{-9}	1.0907×10^{-9}	1.1181×10^{-9}
	0.0001	NA	NA	6.1600×10^{-12}	6.0401×10^{-12}	5.2753×10^{-12}
MSSRE with linear noises $\rho(t) = 6 * (t - 1)$	0.01	NA	NA	4.1569×10^0	4.1569×10^0	4.1569×10^0
	0.001	NA	NA	4.1569×10^0	4.1569×10^0	4.1569×10^0
	0.0001	NA	NA	4.1569×10^0	4.1569×10^0	4.1569×10^0
MSSRE with random noises $\rho(t) \in [99.5, 100.5]$	0.01	NA	NA	1.3161×10^0	1.2095×10^0	1.6278×10^0
	0.001	NA	NA	1.4003×10^0	1.5556×10^0	1.8886×10^0
	0.0001	NA	NA	1.5610×10^0	1.6939×10^0	2.1750×10^0

“NA” means that FSND model (1.15) cannot be used for computing nonstationary QP problem (1.36)

where $J(\theta(t)) \in \mathbb{R}^{3 \times 6}$ is the Jacobian matrix; $\theta(t) \in \mathbb{R}^6$ denotes the joint angle vector of the UR5 robot; $\varpi(t) \in \mathbb{R}^3$ is the joint position vector with $\dot{\varpi}(t)$ its time derivative. The UR5 robot is desired to track a Talbot-shaped trajectory with the execution time of 10 s; g is set as 0.001 s, randomly generated initial positions. The simulation results synthesized by the designed ESEND model (1.13), the constructed CGND model (1.34), and the given EANI model (1.35) are shown as Fig. 1.7, 1.8, 1.9, and 1.10 taking without perturbations and random noises into consideration, where the position error is defined as $\mathbf{e} = \varpi(t) - \varpi_d(t)$ with $\varpi_d(t)$ the desired trajectory. It can be seen from these figures that the designed three models can handle the random noises effectively with the position errors shown in Fig. 1.7c, 1.8, 1.9, and 1.10c all of the order 10^{-4} , which illustrates the successfully completed task and embodies the models with high robustness. In contrast, the tracking task synthesized by Newton iteration (1.3) and discrete-time model (1.16) disturbed by random noises is terrible and failed, as described in Figs. 1.11 and 1.12. Figures 1.11b and 1.12b present that the joint angles of the UR5 robot abruptly fluctuate resulting in nonnegligible position errors and failure of the task. It is worth pointing out that the robot in the tracking process generated by discrete-time model (1.16) is stuck in singularity and the model is invalid in the face of random noises in Fig. 1.12, which even seriously damage robots in practical production. As a continuation, the constructed ESEND model (1.13) and discrete-time model (1.16) are further employed to the UR5 robot to execute the heart-shaped trajectory simulated in virtual robot experimentation platform (V-rep) perturbed by random noises. It can be observed from Figs. 1.13 and 1.14 that the discrete-time model (1.16) shows the bad performance of motion generations and cannot finish the trajectory tracking. The abovementioned experimental results verify the high

Fig. 1.7 Simulation results of UR5 robot tracking the Talbot-shaped trajectory synthesized by the constructed ESEND model (1.13) without perturbations with $g = 0.001$ s. **(a)** Motion trajectory, **(b)** profiles of joint angles, and **(c)** profiles of position error

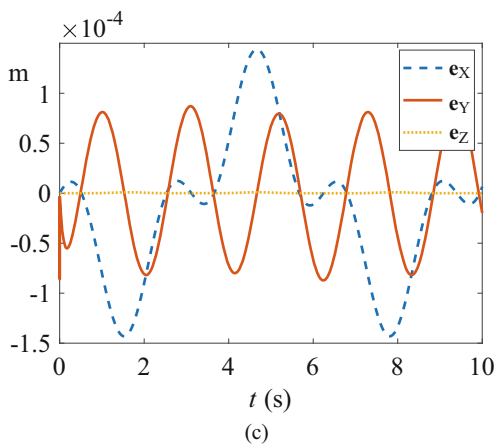
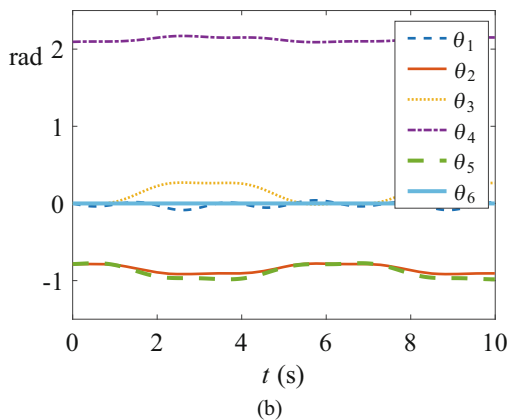
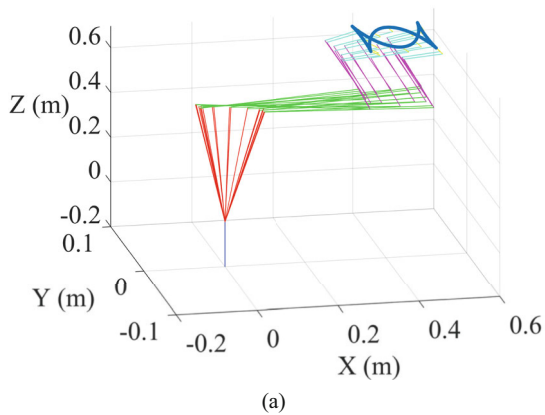


Fig. 1.8 Simulation results of UR5 robot tracking the Talbot-shaped trajectory synthesized by the given ESEND model (1.13) with random noise $\rho(t) \in [1, 10]$ and $g = 0.001$ s. (a) Motion trajectory, (b) profiles of joint angles, and (c) profiles of position error

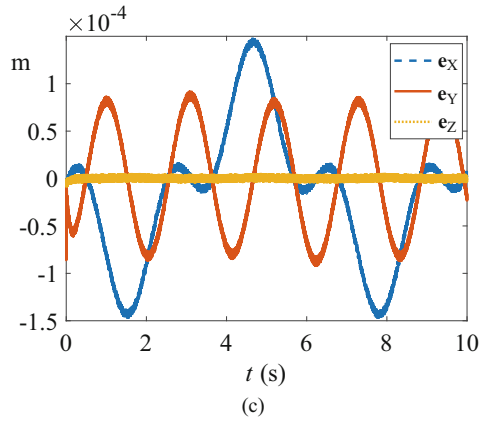
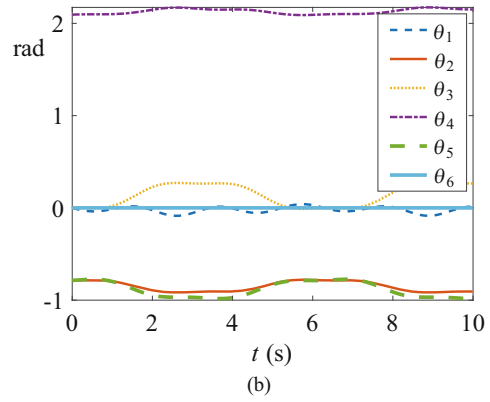
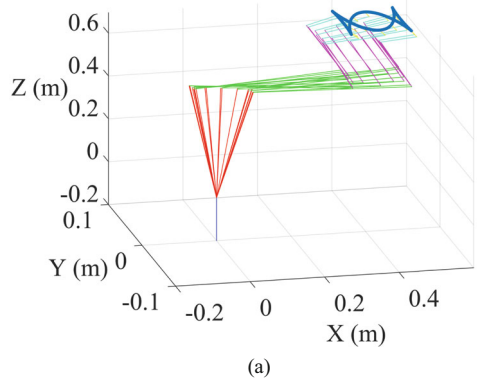


Fig. 1.9 Simulation results of UR5 robot tracking the Talbot-shaped trajectory synthesized by the designed CGND model (1.34) with random noise $\rho(t) \in [1, 10]$ and $g = 0.001$ s. (a) Motion trajectory, (b) profiles of joint angles, and (c) profiles of position error

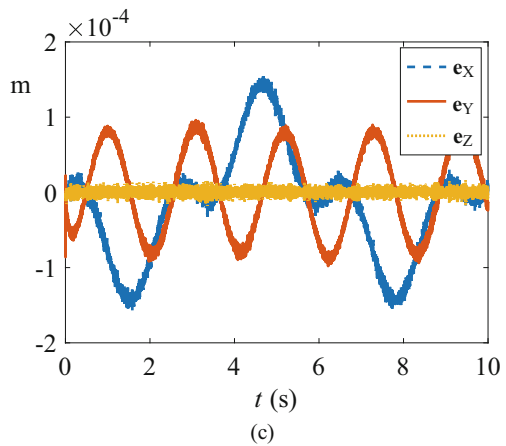
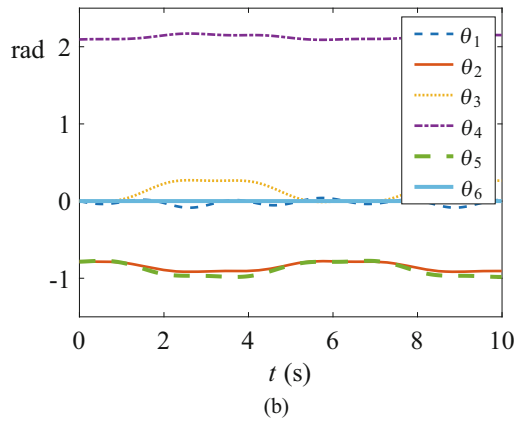
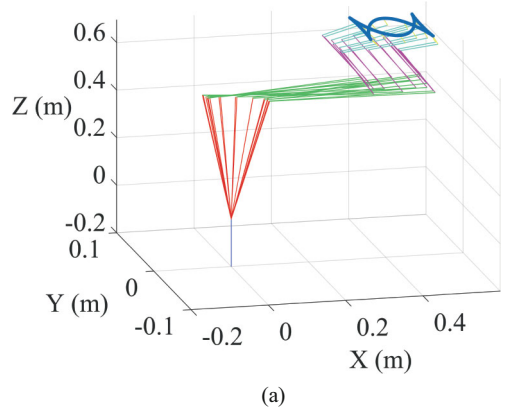


Fig. 1.10 Simulation results of UR5 robot tracking the Talbot-shaped trajectory synthesized by the constructed EANI model (1.35) with random noise $\rho(t) \in [1, 10]$ and $g = 0.001$ s. (a) Motion trajectory, (b) profiles of joint angles, and (c) profiles of position error

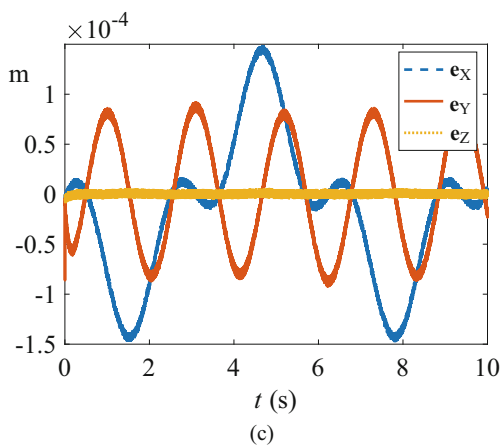
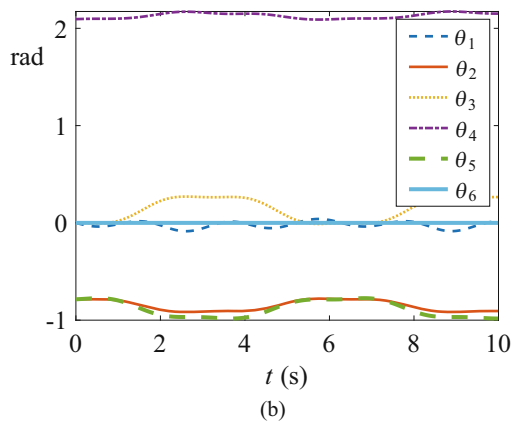
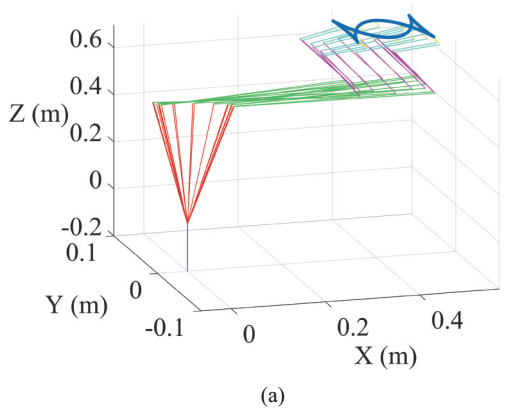


Fig. 1.11 Simulation results of UR5 robot tracking the Talbot-shaped trajectory synthesized by Newton iteration (1.3) with random noise $\rho(t) \in [1, 10]$ and $g = 0.001$ s. **(a)** Motion trajectory, **(b)** profiles of joint angles, and **(c)** profiles of position error

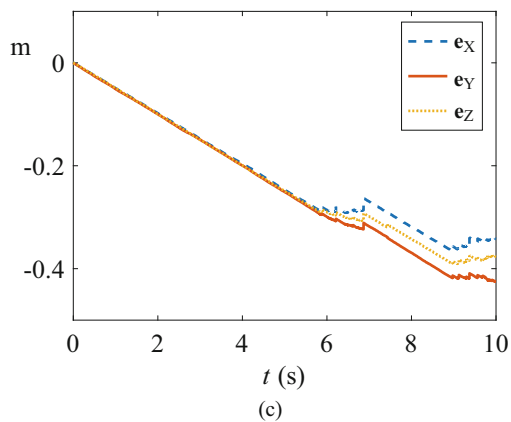
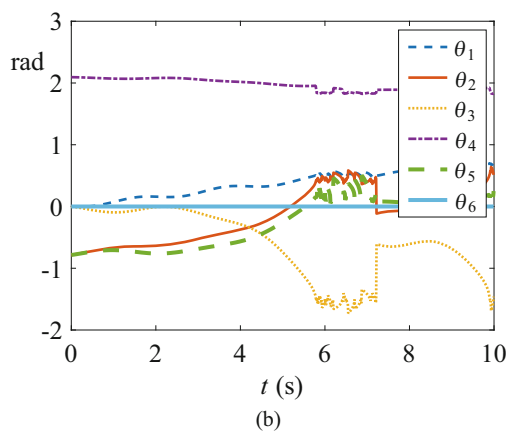
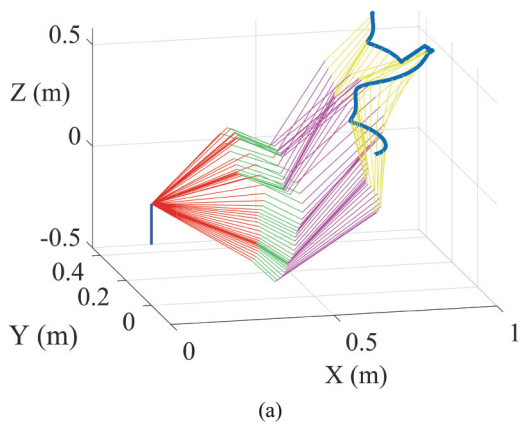


Fig. 1.12 Simulation results of UR5 robot tracking the Talbot-shaped trajectory synthesized by discrete-time model (1.16) with random noise $\rho(t) \in [1, 10]$ and $g = 0.001$ s. **(a)** Motion trajectory, **(b)** profiles of joint angles, and **(c)** profiles of position error

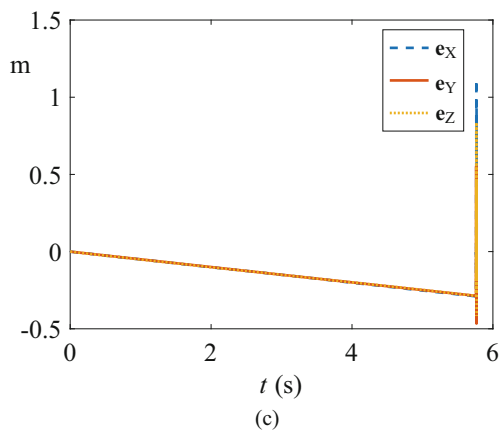
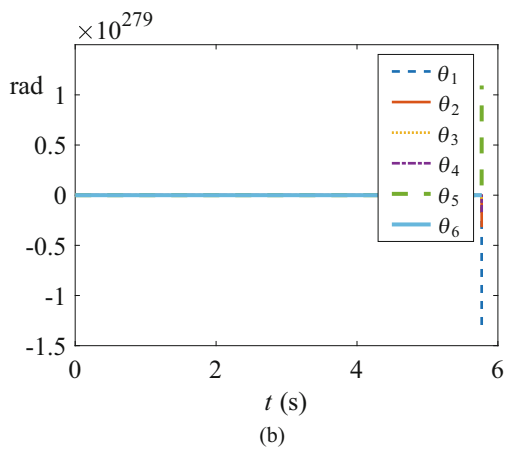
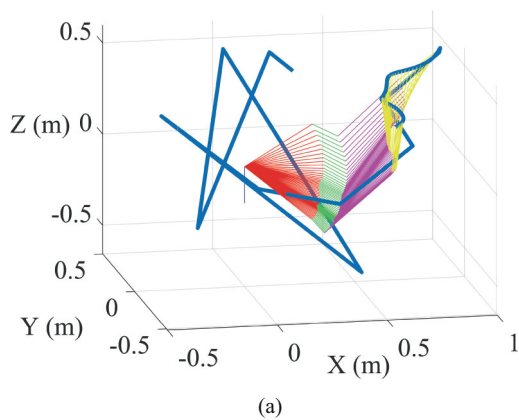


Fig. 1.13 Snapshots of the UR5 robot simulated on the V-rep platform tracking the heart-shaped trajectory synthesized by the given ESEND model (1.13) with random noise $\rho(t) \in [1, 10]$ and $g = 0.001$ s

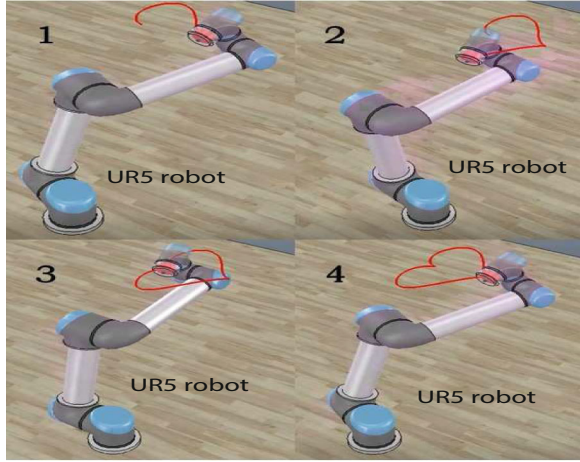
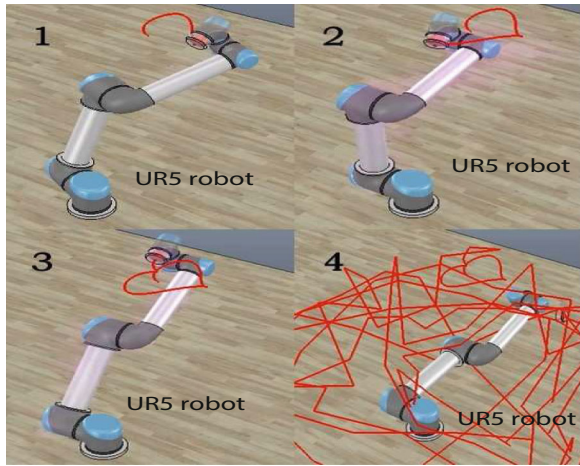


Fig. 1.14 Snapshots of the UR5 robot simulated on the V-rep platform tracking the heart-shaped trajectory synthesized by discrete-time model (1.16) with random noise $\rho(t) \in [1, 10]$ and $g = 0.001$ s



accuracy and robustness of the given ESEND model (1.13), the designed CGND model (1.34), and the constructed EANI model (1.35).

1.5 Extension of the Bounded QP

In this section, a further research direction in QP problems constrained by inequalities and a potential application are presented here to show the powerful solving ability and suitability of the given ESEND model (1.13), the designed CGND model (1.34), and the constructed EANI model (1.35).

1.5.1 Preliminaries

Suppose that an additional inequality constraint is added to the nonstationary QP problem (1.1), thereby generating

$$\begin{aligned} & \text{Minimize} && \mathbf{x}^T(t)H(t)\mathbf{x}(t)/2 + \boldsymbol{\rho}^T(t)\mathbf{x}(t) \\ & \text{Subject to} && \Lambda(t)\mathbf{x}(t) = \mathbf{v}(t) \\ & && D(t)\mathbf{x}(t) \leq \boldsymbol{\vartheta}(t), \end{aligned} \quad (1.38)$$

where $H(t)$, $\boldsymbol{\rho}(t)$, $\Lambda(t)$, and $\mathbf{v}(t)$ are the same as those described in Sect. 1.2.1; $D(t) \in \mathbb{R}^{l \times n}$; $\boldsymbol{\vartheta}(t) \in \mathbb{R}^l$. Then, define a Lagrange-multiplier vector as $\boldsymbol{\sigma}(t) \in \mathbb{R}^l$ related to the inequality constraint in (1.38). To calculate the theoretical solution $\mathbf{x}^*(t)$ to Eq. (1.38), the Lagrange-multiplier vectors $\boldsymbol{\chi}(t)$ and $\boldsymbol{\sigma}(t)$ are required to make the following Karush-Kuhn-Tucker conditions true:

$$\begin{aligned} & H(t)\mathbf{x}(t) + \boldsymbol{\rho}(t) + \Lambda^T(t)\boldsymbol{\chi}(t) + D^T(t)\boldsymbol{\sigma}(t) = \mathbf{0} \\ & \Lambda(t)\mathbf{x}(t) - \mathbf{v}(t) = \mathbf{0} \\ & \boldsymbol{\sigma}^T(t)(\boldsymbol{\vartheta}(t) - D(t)\mathbf{x}(t)) = 0 \\ & \boldsymbol{\sigma}(t) \geq \mathbf{0}, \quad \boldsymbol{\vartheta}(t) - D(t)\mathbf{x}(t) \geq \mathbf{0}. \end{aligned} \quad (1.39)$$

For solving the above nonlinear QP problem, a type of vector-valued nonlinear complementary problem function is leveraged as follows [31]:

$$L(\boldsymbol{\phi}_1, \boldsymbol{\phi}_2) = \sqrt{\boldsymbol{\phi}_1 \bullet \boldsymbol{\phi}_1 + \boldsymbol{\phi}_2 \bullet \boldsymbol{\phi}_2} + \zeta - (\boldsymbol{\phi}_1 + \boldsymbol{\phi}_2),$$

where $\bullet$ stands for the scalar product operation, and $\zeta \rightarrow \mathbf{0}_+$ is the perturbation term. Based on the definition of this function, Eq. (1.39) can be further evolved as

$$\begin{aligned} & H(t)\mathbf{x}(t) + \boldsymbol{\rho}(t) + \Lambda^T(t)\boldsymbol{\chi}(t) + D^T(t)\boldsymbol{\sigma}(t) = \mathbf{0} \\ & \Lambda(t)\mathbf{x}(t) - \mathbf{v}(t) = \mathbf{0} \\ & L(\boldsymbol{\vartheta}(t) - D(t)\mathbf{x}(t), \boldsymbol{\sigma}(t)) = 0, \end{aligned} \quad (1.40)$$

which can be manipulated into the following equation:

$$G(t)\boldsymbol{\xi}(t) - \mathbf{s}(t) = \mathbf{0},$$

with $G(t) = [H(t) \Lambda^T(t) D^T(t); \Lambda(t) \mathbf{0} \mathbf{0}; -D(t) \mathbf{0} \mathbf{I}] \in \mathbb{R}^{(n+m+l) \times (n+m+l)}$, $\boldsymbol{\xi}(t) = [\mathbf{x}(t) \boldsymbol{\chi}(t) \boldsymbol{\sigma}(t)]^T \in \mathbb{R}^{n+m+l}$, and $\mathbf{s}(t) = [\mathbf{s}_1(t) \mathbf{s}_2(t) \mathbf{s}_3(t)]^T \in \mathbb{R}^{n+m+l}$ where $\mathbf{s}_1(t) = -\boldsymbol{\rho}(t)$; $\mathbf{s}_2(t) = \mathbf{v}(t)$; $\mathbf{s}_3(t) = -\boldsymbol{\vartheta}(t) + \sqrt{\boldsymbol{\vartheta}(t) \bullet \boldsymbol{\vartheta}(t) + \boldsymbol{\sigma}(t) \bullet \boldsymbol{\sigma}(t) + \zeta}$; $\boldsymbol{\vartheta}(t) = \boldsymbol{\vartheta}(t) - D(t)\mathbf{x}(t)$.

As a consequence, coping with the nonstationary nonlinear QP problem (1.38) is just to find the theoretical solution to the above equation. Hence, it is evident that the given ESEND model (1.13), the designed CGND model (1.34), and the constructed EANI model (1.35) are able to be employed to solve the nonstationary nonlinear QP problem (1.38).

1.5.2 Application in κ -WTA

Hereinafter, an application in κ -winner take-all (κ -WTA) operation is offered, which can be converted into the form of the nonlinear QP problem (1.38). The κ -WTA operation, as a special case of the winner take-all operation that is screening the largest data from a series of dataset, is aimed at getting the maximum κ -values from the input in real time. The mathematical expression of the κ -WTA operation is

$$\mathbf{x}_i = \Phi(\rho_i) = \begin{cases} 1, & \text{for } \rho_i \in \{\kappa \text{ largest elements of } \rho\} \\ 0, & \text{for others.} \end{cases}$$

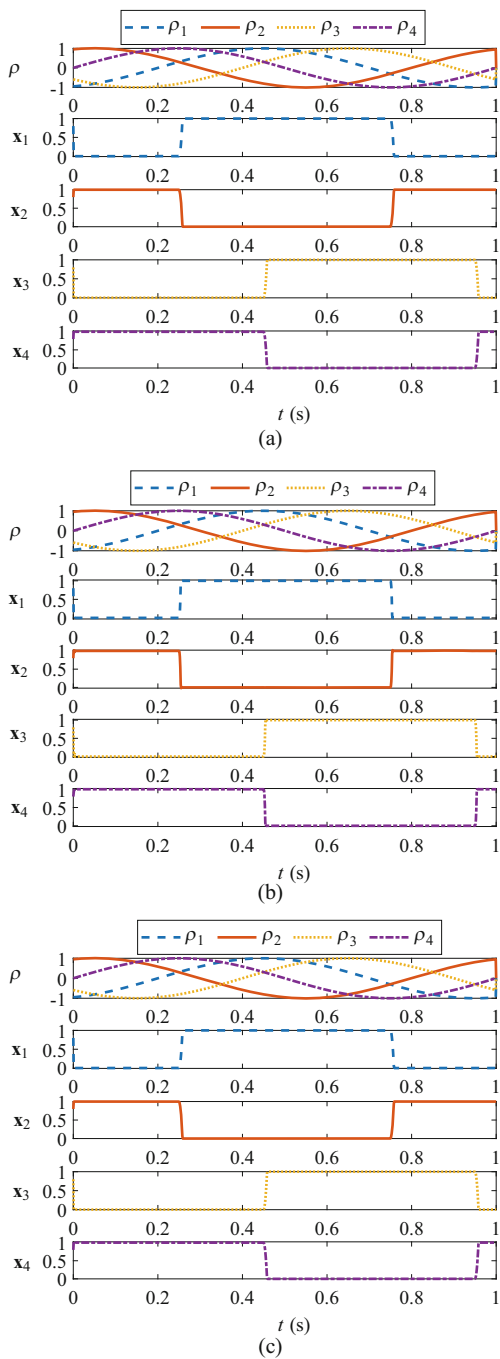
The above equation is demonstrated to be equal to a QP problem with an inequality constraint described as follows [32, 33]:

$$\begin{aligned} & \text{Minimize} && \mathcal{E} \mathbf{x}^T \mathbf{x} - \rho^T \mathbf{x} \\ & \text{Subject to} && \mathcal{F}^T \mathbf{x} = \kappa \\ & && 0 \leq \mathbf{x}_i \leq 1, \quad i \in [1, n] \end{aligned} \tag{1.41}$$

where $0 < \mathcal{E} < 0.5(\check{\rho}_\kappa - \check{\rho}_{\kappa+1}) \in \mathbb{R}$ with $\check{\rho}_\kappa$ and $\check{\rho}_{\kappa+1}$ being the κ th and $(\kappa + 1)$ th largest data of ρ , respectively; $\mathcal{F} = [1, 1, \dots, 1]^T \in \mathbb{R}^n$; κ is the number of maximums of $\mathbf{x}$ to be output. In consequence, to display the outcome of the κ -WTA operation and illustrate the effectiveness of the given ESEND model (1.13), the designed CGND model (1.34), and the constructed EANI model (1.35), an instructive example is conducted next.

Give a set of sinusoidal functions as input data $\rho_i = -\sin(2\pi(t + 0.4(i + 1)))$, $i = 1, 2, 3, 4$; $\kappa = 2$; $\mathcal{E} = 10^{-3}$. The simulation results on the κ -WTA operation (1.41) synthesized by the given ESEND model (1.13), the designed CGND model (1.34), and the constructed EANI model (1.35) with the sampling gap $g = 0.0001$ s are portrayed in Fig. 1.15. It is found clearly that the three given models can all select the two largest data from the time-dependent input signals timely, with smooth picking process and satisfactory results. The application in κ -WTA operation highlights the superiority and efficiency of the designed ESEND model (1.13), the constructed CGND model (1.34), and the given EANI model (1.35) for computing nonstationary QP problems and proves their practicability and portability in industry.

Fig. 1.15 Simulation results for solving the κ -WTA operation (1.41) with the determined values. (a) Inputs and outputs of the given ESEND model (1.13), (b) inputs and outputs of the designed CGND model (1.34), and (c) inputs and outputs of the constructed EANI model (1.35)



1.6 Conclusion

It is worth mentioning that it is the first time to propose an error-sum enhanced neural dynamics model (1.13) with superior robustness against various kinds of noises from the control perspective on nonstationary QP problems solving. The theoretical analyses have illustrated that the designed ESEND model (1.13) has the global convergence property and high robustness. From the viewpoint of control, a systematic framework has been built that first summarizes and interprets the existing models for computing nonstationary QP problems. By means of the systematic framework, the control technology has been imparted to the classical Newton iteration and gradient-based neural network model to construct another two models (1.34) and (1.35). Moreover, for comparison purpose, Newton iteration (1.3), discrete-time model (1.16), and TGBNN model (1.17) have been employed for the same solving task with the corresponding experimental results provided. In view of potential applications in science and industry, the robot tracking control and κ -WTA operation have been investigated by exploiting the constructed ESEND model, the given CGND model, and the designed EANI model. All the results have illustrated the superiority and validity of the constructed models, and to a large extent this chapter provides a theoretical basis for the future research or improvement of algorithms/models in the control domain.

References

1. Bhaya A, Kaszkurewicz E (2007) A control-theoretic approach to the design of zero finding numerical methods. *IEEE Trans Autom Control* 56(2):1014–1026
2. Liu M, Wu H, Shi Y, Jin L (2023) High-order robust discrete-time neural dynamics for time-varying multilinear tensor equation with $\mathcal{M}$ -tensor. *IEEE Trans Ind Inform* 19(9):9457–9467
3. Bhaya A, Kaszkurewicz E (2006) Control perspectives on numerical algorithms and matrix problems. SIAM, Philadelphia
4. Jin L, Zhang Y, Li S, Zhang Y (2017) Noise-tolerant ZNN models for solving time-varying zero-finding problems: a control-theoretic approach. *IEEE Trans Autom Control* 62(2):902–907
5. Ahmadi-Moshkenani P, Johansen TA, Olaru S (2018) Combinatorial approach toward multiparametric quadratic programming based on characterizing adjacent critical regions. *IEEE Trans Autom Control* 63(10):3221–3231
6. Bemporad A (2018) A numerically stable solver for positive semidefinite quadratic programs based on nonnegative least squares. *IEEE Trans Autom Control* 63(2):525–531
7. Saraf N, Bemporad A (2020) A bounded-variable least-squares solver based on stable QR updates. *IEEE Trans Autom Control* 65(3):1242–1247
8. Guo D, Nie Z, Yan L (2017) Novel discrete-time Zhang neural network for time-varying matrix inversion. *IEEE Trans Syst Man Cybern Syst* 47(8):2301–2310
9. Jin L, Li S, Hung L, Luo X (2017) Manipulability optimization of redundant manipulators using dynamic neural networks. *IEEE Trans Ind Electron* 64(6):4710–4720
10. Zhang Y, Li S, Liu X (2017) Adaptive near-optimal control of uncertain systems with application to underactuated surface vessels. *IEEE Trans Control Syst Technol* 26(4):1204–1218

11. Liufu Y, Jin L, Li S (2023) Modified gradient projection neural network for multiset constrained optimization. *IEEE Trans Ind Inform* 19(9):9413–9423
12. Yang Y, Ding SX (2012) A control-theoretic study on iterative solutions to nonlinear equations for applications in embedded systems. *Automatica* 48(4):583–594
13. Hasan A, Kerrigan EC, Constantinides GA (2013) Control-theoretic forward error analysis of iterative numerical algorithms. *IEEE Trans Autom Control* 58(6):1524–1529
14. Wang J (1992) Recurrent neural network for solving quadratic programming problems with equality constraints. *Electron Lett* 28(14):1345–1347
15. Zhang Y, Wang J (2002) Global exponential stability of recurrent neural networks for synthesizing linear feedback control systems via pole assignment. *IEEE Trans Neural Netw* 13(3):633–644
16. Yan J, Jin L, Yuan Z, Liu Z (2022) RNN for receding horizon control of redundant robot manipulators. *IEEE Trans Ind Electron* 69(2):1608–1619
17. Sun C, Ye M, Hu G (2017) Distributed time-varying quadratic optimization for multiple agents under undirected graphs. *IEEE Trans Autom Control* 62(7):3687–3694
18. Zhang Z, Lu Y, Zheng L, Li S, Yu Z, Li Y (2018) A new varying-parameter convergent-differential neural-network for solving time-varying convex QP problem constrained by linear-equality. *IEEE Trans Autom Control* 63(12):4110–4125
19. Liao-McPherson DL, Huang M, Kolmanovsky I (2019) A regularized and smoothed Fischer-Burmeister method for quadratic programming with applications to model predictive control. *IEEE Trans Autom Control* 64(7):2937–2944
20. Galeani S, Menini L, Potini A (2012) Robust trajectory tracking for a class of hybrid systems: an internal model principle approach. *IEEE Trans Autom Control* 57(2):344–359
21. Zhang Y, Li Z (2009) Zhang neural network for online solution of time-varying convex quadratic program subject to time-varying linear-equality constraints. *Phys Lett A* 373(18/19):1639–1643
22. Jin L, Zhang Y, Li S, Zhang Y (2016) Modified ZNN for time-varying quadratic programming with inherent tolerance to noises and its application to kinematic redundancy resolution of robot manipulators. *IEEE Trans Ind Electron* 63(11):6978–6988
23. Guo D, Nie Z, Yan L (2016) Theoretical analysis, numerical verification and geometrical representation of new three-step DTZD algorithm for time-varying nonlinear equations solving. *Neurocomputing* 214:516–526
24. Liao B, Zhang Y, Jin L (2016) Taylor $O(h^3)$ discretization of ZNN models for solving dynamic equality-constrained quadratic programming with application to manipulators. *IEEE Trans Neural Netw Learn Syst* 27(2):225–267
25. Zhang Y, Mu B, Zheng H (2013) Member, “Link between and comparison and combination of Zhang neural network and quasi-Newton BFGS method for time-varying quadratic minimization,” *IEEE Trans Cybern* 43(2):490–503
26. Li S, Zhou M, Luo X (2018) Modified primal-dual neural networks for motion control of redundant manipulators with dynamic rejection of harmonic noises. *IEEE Trans Neural Netw Learn Syst* 29(10):4791–4801
27. Bouyarmane K, Kheddar A (2018) On weight-prioritized multitask control of humanoid robots. *IEEE Trans Autom Control* 63(6):1632–1647
28. Wang H, Ren W, Cheah CC, Xie Y, Lyu S (2020) Dynamic modularity approach to adaptive control of robotic systems with closed architecture. *IEEE Trans Autom Control* 65(6):2760–2767
29. Chen D, Zhang Y (2018) Robust zeroing neural-dynamics and its time-varying disturbances suppression model applied to mobile robot manipulators. *IEEE Trans Neural Netw Learn Syst* 29(9):4385–4397
30. Xie Z, Li S, Jin L (2024) A bi-criteria kinematic strategy for motion/force control of robotic manipulator. *IEEE Trans Autom Sci Eng*. In Press. <https://doi.org/10.1109/TASE.2023.3313564>

31. Nazemi A, Nazemi M (2014) A gradient-based neural network method for solving strictly convex quadratic programming problems. *Cogn Comput* 6(3):484–495
32. Li S, Zhou M, Luo X, You Z (2017) Distributed winner-take-all in dynamic networks. *IEEE Trans Autom Control* 62(2):577–589
33. Liu S, Wang J (2006) A simplified dual neural network for quadratic programming with its KWTa application. *IEEE Trans Neural Netw* 17(6):1500–1510

Chapter 2

Complex-Valued Discrete-Time Neural Dynamics



2.1 Introduction

Quadratic programming (QP) problems are frequently encountered in various spheres, such as motion planning of robots [1], image processing [2], and signal processing [3]. For example, the redundancy resolution problems existing in robot manipulators can be formulated as a QP problem and solved by the corresponding solvers [4]. In recent decades, complex-valued problems have received much attention due to their wide applications, such as electric circuit design and analysis [5], signal analysis and processing [6], and feedback control of dynamics systems [7]. The shift from real domain to complex domain motivates researchers to devote their effort on the solution of complex-valued problems. Especially, a great deal of computational models have been presented in recent decades. In [8], Liao et al. extend the static QP problems from the real domain to complex domain with the real and imaginary components of static complex QP problems solved in the real domain separately, which increases the unnecessary complexities and may lose essential insights on the problems structure. Evidently, for a complex QP problem, it is more convenient to directly analyze and cope with it in a complex field. It is worth noting that the Newton iteration method and its modifications [9] are traditional approaches for solving static QP problems.

However, it is deserved to mention that many practical problems are inherently dynamic [10, 11]. For time-dependent applications requiring a rapid convergence to theoretical solutions, traditional methods may fail to track the theoretical solution of a time-dependent problem due to the lack of velocity compensation of dynamic parameters [12–19]. Therefore, large residuals may be generated for the solution of time-dependent problems synthesized by the conventional methods. It has been verified in [20] that the traditional methods cannot deal with a time-dependent system of linear equations in a predictive manner, resulting in non-negligible

residuals. In addition, a large number of recurrent neural network (RNN) models are presented and investigated, which can cope with many mathematical and engineering problems with time-dependent coefficients, including the QP problems [21, 22]. A discrete-time RNN is presented to solve QP problems with linear constraints in [22], of which the global exponential stability is proved and the structure can be implemented effectively by using very-large-scale integration technology. Thus, RNNs are generally considered as a powerful computing tool, which can solve various challenging problems occurring in numerous scientific and engineering fields online [23–27]. Especially, the zeroing neural network (ZNN), as a type of RNN, can accurately track the time-dependent solutions by exploiting the time derivatives of time-dependent parameters [28]. One of the recent achievements on time-dependent problem-solving is the Taylor-type discrete-time zeroing neural network models introduced in [29], which are superior to other models in terms of accuracy. However, these models in [29] are sensitive to noises and perturbations, and a slight noise would degrade their performance significantly, which indicates that new progress along the denoising direction should be made.

During the solving process of QP problems, noises and perturbations are inevitable, which often leads to the performance degradation of a computational model or even the failure of the solving task. However, the measures for denoising are rarely incorporated into the design of computational models [30], which severely restricts the applications in complex environments. Zhang and Xia introduce a complex-valued conjugate gradient projection (CCGP) method for solving complex optimization problems with linear equality constraints [31], which is of global convergence without considering the noises. Discrete ZNN models given in [4] can solve time-dependent system of linear equations effectively and accurately with matrix inversion operation, which are analyzed and employed in the real domain. Moreover, given that a dynamical system depicted in continuous form cannot be implemented directly on digital computers, it is worth exploiting the discrete-time neural dynamics models to solve the complex-valued time-dependent QP problems. In addition, the matrix inversion operation is always involved in the online solving of QP problems. However, for many complicated problems, it is difficult to accurately calculate the inverse of the matrix. Given the facts that complex-valued QP problems are frequently encountered in scientific and engineering areas, that there are few methods for solving complex-valued time-dependent QP problems, and that noises and perturbations degrade the solving performance significantly, it is desirable to propose new models to remedy the weaknesses of the existing solutions. Therefore, in this chapter, two new complex-valued discrete-time neural dynamics (DTND) models are designed and investigated for solving perturbed time-dependent complex QP problems, of which one model uses quasi-Newton Broyden-Fletcher-Goldfarb-Shanno (BFGS) to eliminate the matrix inversion.

The remainder of this chapter is organized into seven sections. Section 2.2 presents the scheme formulation and provides some preliminaries. In Sect. 2.3, by defining a complex-valued error function, the complex-valued DTND and

DTND-BFGS models are constructed and investigated for the time-dependent QP problem in complex domain. Theoretical analyses are further presented in Sect. 2.4 to illustrate the convergence property of the presented models and the superior performance in the presence of perturbations. In addition, Sect. 2.5 provides numerical simulation results to conduct comparisons. Moreover, applications to the robot manipulator and FIR filter design are shown in Sect. 2.6. Finally, conclusions are conducted in Sect. 2.7. The main contributions of this chapter are summarized as follows.

- This chapter focuses on solving the time-dependent complex QP problem in the presence of noises, which is quite different from the existing research on continuous-time dependent problems or in the real domain.
- Two complex-valued DTND models are given, and their effectiveness and global convergence performance are analyzed and presented. Besides, the superior property of noise suppressing is verified.
- Numerical experiments results are presented, which demonstrate the superiority of the designed complex-valued DTND models by comparing with the traditional methods in the presence of various noises.
- Two application examples are provided to illustrate the efficacy and superiority of the complex-valued DTND models for solving the perturbed time-dependent complex QP problems.

2.2 Preliminary and Scheme Formulation

In this chapter, we are concerned with the time-dependent complex quadratic programming problem with the linear-equality constraint shown as follows:

$$\text{Minimize} \quad \|Q(t)\mathbf{y}(t) + \mathbf{q}(t)\|^2, \quad (2.1)$$

$$\text{subject to} \quad D(t)\mathbf{y}(t) = \mathbf{a}(t), \quad (2.2)$$

where $\|\cdot\|$ denotes the 2-norm; $\mathbf{y}(t)$ is an m -dimensional time-dependent complex vector; $Q(t) \in \mathbb{C}^{l \times m}$ and $D(t) \in \mathbb{C}^{n \times m}$ (being of full rank) are coefficient matrices; and the coefficient vectors $\mathbf{q}(t) \in \mathbb{C}^l$, $\mathbf{a}(t) \in \mathbb{C}^n$. In addition, some definitions on optimization in the complex domain are introduced as follows to lay a basis for further research.

Given a real-valued function $f(\mathbf{y})$ with a complex-valued variable $\mathbf{y}$, let $\mathbf{y} = \mathbf{c} + i\mathbf{d}$, i.e., the real and the imaginary parts of the complex variable are written respectively. Then $f(\mathbf{y})$ can be reformulated as a real bivariate function $h(\mathbf{c}, \mathbf{d})$. According to Brandwood's theorem introduced in [31], the following instruction is provided.

The Cauchy-Riemann (C-R) conditions are not applicable in complex variable optimization problems so that the objective functions are not analytic. Thus, Brandwood's analytic theory is applied here to solve the difficulty. That is, $f(\mathbf{y})$ is the analysis of the complex-valued variable $\mathbf{y}$ and its complex conjugate twin $\mathbf{y}^*$, where $\mathbf{y}$ and $\mathbf{y}^*$ are deemed as independent variables if $f(\mathbf{y})$ satisfies Brandwood's analytic theory. A formula based on the above description is shown as

$$\nabla f(\mathbf{y}) = \frac{\partial f}{\partial \mathbf{y}^*} = \frac{1}{2} \left(\frac{\partial h}{\partial \mathbf{c}} + i \frac{\partial h}{\partial \mathbf{d}} \right). \quad (2.3)$$

For further analyses, let $f(\mathbf{y}) = \|Q(t)\mathbf{y}(t) + \mathbf{q}(t)\|^2$, $N(t) = Q^H(t)Q(t)$, $\mathbf{b}(t) = Q^H(t)\mathbf{q}(t)$. The $Q^H(t)$ denotes the conjugate transpose of $Q(t)$. Then $f(\mathbf{y})$ can be redefined as

$$f(\mathbf{y}) = \mathbf{y}^H(t)N(t)\mathbf{y}(t) + \mathbf{b}^H(t)\mathbf{y}(t) + \mathbf{y}^H(t)\mathbf{b}(t) + \mathbf{q}^H(t)\mathbf{q}(t), \quad (2.4)$$

where the coefficient vector $\mathbf{b}(t) \in \mathbb{C}^m$, Hessian matrix $N(t) \in \mathbb{R}^{m \times m}$, which is semi-definite or positive definite if $Q(t)$ is of full rank. By Eq. (2.3), we have

$$\nabla f(\mathbf{y}) = Q^H(t)Q(t)\mathbf{y}(t) + Q^H(t)\mathbf{q}(t) = N(t)\mathbf{y}(t) + \mathbf{b}(t).$$

The complex quadratic programming problem with linear equality constrain can be rearranged as

$$\begin{aligned} \text{Minimize} \quad & f(\mathbf{y}) = \|Q(t)\mathbf{y}(t) + \mathbf{q}(t)\|^2, \\ \text{Subject to} \quad & D(t)\mathbf{y}(t) = \mathbf{a}(t). \end{aligned} \quad (2.5)$$

The vectors or matrices mentioned above are time-dependent and time instant $t \in [t_0, t_f]$ can be expressed as $t = kg$ with g denoting the sampling interval and $k=0, 1, 2, \dots$ representing the updating index. Due to the time-dependent characteristics of each variable, the optimal solution $\mathbf{y}(t)$ is time-dependent. To ensure the uniqueness of the solution, Hessian matrix $N(t)$ is restricted to be a positive definite matrix at any time instant t . It is emphasized that the solution $\mathbf{y}((k+1)g)$ should be calculated before the time instant $t = (k+1)g$, while the data $Q((k+1)g)$, $D((k+1)g)$, $\mathbf{a}((k+1)g)$, and $\mathbf{q}((k+1)g)$ are not attainable till the time instant $t = (k+1)g$. In this sense, what we do is a calculation or forecast of the future solution, just using the current and/or former data.

The Lagrange-multiplier vector is utilized to solve QP problem (2.5) in the complex domain. Therefore, its related Lagrangian is shown as follows:

$$L(\mathbf{y}(t), \boldsymbol{\varpi}(t), t) = f(\mathbf{y}) - 2\text{Re}((D(t)\mathbf{y}(t) - \mathbf{a}(t))^H \boldsymbol{\varpi}(t)), \quad (2.6)$$

where $\boldsymbol{\varpi}(t) \in \mathbb{C}^n$ denotes the complex Lagrange-multiplier vector. Then, to get the stagnation point at each moment, $L(\mathbf{y}(t), \boldsymbol{\varpi}(t), t)$ should satisfy the following conditions:

$$\begin{cases} \frac{\partial L(\mathbf{y}(t), \boldsymbol{\varpi}(t), t)}{\partial \mathbf{y}(t)} = \nabla f(\mathbf{y}) + \boldsymbol{\varpi}(t) D^H(t) = 0, \\ \frac{\partial L(\mathbf{y}(t), \boldsymbol{\varpi}(t), t)}{\partial \boldsymbol{\varpi}(t)} = D(t) \mathbf{y}(t) - \mathbf{a}(t) = 0. \end{cases} \quad (2.7)$$

The above equations can be rewritten as

$$S(t) \mathbf{z}(t) = \mathbf{q}(t), \quad (2.8)$$

where

$$S(t) = \begin{bmatrix} N(t) & D^H(t) \\ D(t) & 0_{n \times n} \end{bmatrix} \in \mathbb{C}^{(m+n) \times (m+n)},$$

$$\mathbf{z}(t) = \begin{bmatrix} \mathbf{y}(t) \\ \boldsymbol{\varpi}(t) \end{bmatrix} \in \mathbb{C}^{m+n}, \quad \mathbf{q}(t) = \begin{bmatrix} -\mathbf{b}(t) \\ \mathbf{a}(t) \end{bmatrix} \in \mathbb{C}^{m+n}.$$

The conditions that $N(t)$ is a positive definite matrix, $D(t)$ is of full row rank, and $S(t)$ is a nonsingular matrix at any time instant t are required to fulfill simultaneously, which guarantees the uniqueness of the theoretical solution. According to the above formula, we reformulate the theoretical solution as

$$\mathbf{z}^*(t) = [\mathbf{y}^{*H}(t), \boldsymbol{\varpi}^{*H}(t)]^H = S^\dagger(t) \mathbf{q}(t) \in \mathbb{C}^{m+n}, \quad (2.9)$$

where superscript † denotes the pseudoinversion operation. This theoretical solution $\mathbf{z}^*(t)$ is used to prove the validity of the constructed models. It is worth pointing out that, during the practical computing, $\mathbf{z}_{k+1}^*$ should be obtained during the time interval $[t_k, t_{k+1})$, at which S_{k+1} and $\mathbf{q}_{k+1}$ are unknown.

2.3 Discrete-Time Neural Dynamics Models

In this section, the design procedures of the DTND model and DTND-BFGS model are presented for solving the online time-dependent complex QP problem.

2.3.1 Complex-Valued DTND Model

To solve the time-dependent complex QP problem, an error function is firstly defined as follows:

$$\boldsymbol{\zeta}(t) = S(t) \mathbf{z}(t) - \mathbf{q}(t) \in \mathbb{C}^{m+n}, \quad (2.10)$$

and a definition of error function $\xi(t)$ in real domain is given in [18, 32], which is shown as

$$\dot{\xi}(t) = -\gamma \xi(t) - \lambda \int_0^t \xi(\delta) d\delta. \quad (2.11)$$

In what follows, the above equation is extended to complex domain. Substituting Eq. (2.10) into Eq. (2.11) yields a neural dynamics model for solving the complex QP problem:

$$S(t)\dot{\mathbf{z}}(t) = -\dot{S}(t)\mathbf{z}(t) + \dot{\mathbf{q}}(t) - \gamma(S(t)\mathbf{z}(t) - \mathbf{q}(t)) - \lambda \int_0^t \xi(\delta) d\delta. \quad (2.12)$$

To simplify the expression of the model (2.12), the Euler forward difference formula is used to discretize the continuous neural dynamics, which implements the discretization simply and clearly:

$$\dot{\mathbf{z}}_k = (\mathbf{z}_{k+1} - \mathbf{z}_k)/g. \quad (2.13)$$

Therefore, discretizing the formula (2.12) leads to the following DTND model:

$$\begin{aligned} \mathbf{z}_{k+1} = & \mathbf{z}_k + S_k^\dagger (-(S_k - S_{k-1})\mathbf{z}_k + \mathbf{q}_k - \mathbf{q}_{k-1} - \\ & \gamma g(S_k \mathbf{z}_k - \mathbf{q}_k) - \lambda g \sum_{i=0}^k (S_i \mathbf{z}_i - \mathbf{q}_i)). \end{aligned} \quad (2.14)$$

Considering the impact of noises and perturbations on theoretical solution of the complex-valued DTND model, the above equation can be rewritten as follows:

$$\begin{aligned} \mathbf{z}_{k+1} = & \mathbf{z}_k + S_k^\dagger (-(S_k - S_{k-1})\mathbf{z}_k + \mathbf{q}_k - \mathbf{q}_{k-1} - \\ & \Delta(S_k \mathbf{z}_k - \mathbf{q}_k) - \Lambda \sum_{i=0}^k (S_i \mathbf{z}_i - \mathbf{q}_i) + \varphi(t)), \end{aligned} \quad (2.15)$$

where $\Delta = \gamma g$, $\Lambda = \lambda g$ and $\varphi(t)$ denotes the measurement noises. To better demonstrate the performance of the model presented in this chapter, three existing models are listed here for theoretical verification and simulation comparison.

The traditional discrete-time ZNN model based on the continuous-time ZNN model introduced in [33] is provided here

$$\mathbf{z}_{k+1} = \mathbf{z}_k + S_k^\dagger (-g\dot{S}_k \mathbf{z}_k + g\dot{\mathbf{q}}_k - \Delta(S_k \mathbf{z}_k - \mathbf{q}_k)). \quad (2.16)$$

The traditional Newton iteration method for solving (2.5) is formulated as

$$\mathbf{z}_{k+1} = \mathbf{z}_k - S_k^\dagger (S_k \mathbf{z}_k - \mathbf{q}_k) = S_k^\dagger \mathbf{q}_k. \quad (2.17)$$

The detailed descriptions on the CCGP algorithm [31] are briefly described as follows:

Algorithm 1 CCGP

1. Initial feasible point $\mathbf{y}_0$ is set as $\mathbf{y}_0 = D^\dagger \mathbf{a}$ and $\mathcal{E} > 0$.
2. Calculate $\nabla f_0 = N\mathbf{y}_0 + \mathbf{b}$ and let $\mathbf{q}_0 = \ell_0 = -(I_m - D^\dagger D)\nabla f_0$.
3. Calculate

$$\begin{pmatrix} \mathbf{y}_k \\ \nabla f_k \end{pmatrix} = \begin{pmatrix} \mathbf{y}_{k-1} \\ \nabla f_{k-1} \end{pmatrix} + \iota_{k-1} \begin{pmatrix} \mathbf{q}_{k-1} \\ N\mathbf{q}_{k-1} \end{pmatrix} \quad (2.18)$$

where $\iota_{k-1} = \|\ell_{k-1}\|^2 / (\mathbf{q}_{k-1}^H N \mathbf{q}_{k-1})$ denotes the step size.

4. Calculate $\mathbf{q}_k = -(I_m - D^\dagger D)\nabla f_k$. When $\|\ell_k\| \leq \mathcal{E}$, the output is $\mathbf{y}_k$. Otherwise, update $\mathbf{q}_k = \ell_k + \beta_{k-1} \mathbf{q}_{k-1}$, where $\beta_{k-1} = \|\ell_k\|^2 / \|\ell_{k-1}\|^2$. Then let $k := k + 1$ and go to 3.
-

2.3.2 Complex-Valued DTND-BFGS Model

For complicated QP problems, computing the inverse of a complex-valued time-dependent matrix usually consumes much time and may not guarantee high efficiency and sufficient precision of the solution. Therefore, it is necessary to eliminate the matrix inversion operation. It should be noted that if the Hessian matrix is of simple structure, it is possible to directly obtain the inverse matrix $S^\dagger(t)$ by using the DTND model, whereas the DTND-BFGS method can be adopted. Thus, exploiting the quasi-Newton Broyden-Fletcher-Goldfarb-Shanno (BFGS) method to modify the DTND model (2.14) yields [34–36]

$$\begin{aligned} \mathbf{z}_{k+1} = & \mathbf{z}_k + \tilde{U}_k (-(S_k - S_{k-1})\mathbf{z}_k + \mathbf{q}_k - \mathbf{q}_{k-1} - \\ & \gamma g(S_k \mathbf{z}_k - \mathbf{q}_k) - \lambda g \sum_{i=0}^k (S_i \mathbf{z}_i - \mathbf{q}_i)), \end{aligned} \quad (2.19)$$

where $\tilde{U}_k$ is an approximation of $S_k^\dagger$ which is obtained from the previous $\tilde{U}_{k-1}$ through iteration by using the BFGS method. Equation (2.19) is termed the complex-valued DTND-BFGS model. The iterative relationship between $\tilde{U}_k$ and $\tilde{U}_{k+1}$ is formulated as

$$\tilde{U}_{k+1} = \tilde{U}_k + \Delta \tilde{U}_k \quad (k = 0, 1, 2, 3, \dots), \quad (2.20)$$

where $\Delta \tilde{U}_k = (\varsigma_k^H \kappa_k + \kappa_k^H \tilde{U}_k \kappa_k)(\varsigma_k \varsigma_k^H) / (\varsigma_k^H \kappa_k)^2 - (\tilde{U}_k \kappa_k \varsigma_k^H + \varsigma_k \kappa_k^H \tilde{U}_k) / (\varsigma_k^H \kappa_k)$ is a matrix that represents the accumulation item, of which $\varsigma_k = \mathbf{z}_{k+1} - \mathbf{z}_k$ and

$\kappa_k = S_k \varsigma_k$ [34–36]. In order to ensure the accuracy and convergence performance, $\tilde{U}_0$ can be simply set to $S^\dagger(\mathbf{z}_0, t_0)$.

2.3.3 Model Simplification

To facilitate further analyses on the given models, a simpler form of the neural dynamics model (2.14) is presented here by means of the definition of the error function and the following theorem.

Theorem 2.1 *The complex-valued DTND model (2.14) can be reformulated as*

$$(1 + \Delta + \Lambda)\boldsymbol{\zeta}_k - \boldsymbol{\zeta}_{k-1} + \Lambda \sum_{i=0}^{k-1} \boldsymbol{\zeta}_i + \mathbf{O}(g^2) = 0,$$

where $\mathbf{O}(g^2)$ denotes a vector composed of $O(g^2)$.

Proof The complex-valued DTND model (2.14) can be rewritten as

$$S_k(\mathbf{z}_{k+1} - \mathbf{z}_k) = \mathbf{q}_k - \mathbf{q}_{k-1} - (S_k - S_{k-1})\mathbf{z}_k - \Delta\boldsymbol{\zeta}_k - \Lambda \sum_{i=0}^k \boldsymbol{\zeta}_i. \quad (2.21)$$

The above equation can be simplified as

$$S_k\mathbf{z}_{k+1} = \mathbf{q}_k - \mathbf{q}_{k-1} + S_{k-1}\mathbf{z}_k - \Delta\boldsymbol{\zeta}_k - \Lambda \sum_{i=0}^k \boldsymbol{\zeta}_i. \quad (2.22)$$

Applying the Taylor expansion expression to the variable in the above equation, we can obtain

$$\mathbf{z}_{k+1} = \mathbf{z}_k + g\dot{\mathbf{z}}_k + \mathbf{O}(g^2), \quad (2.23)$$

and

$$\mathbf{z}_{k-1} = \mathbf{z}_k - g\dot{\mathbf{z}}_k + \mathbf{O}(g^2), \quad (2.24)$$

of which the latter can be rewritten as

$$\mathbf{z}_k = \mathbf{z}_{k-1} + g\dot{\mathbf{z}}_k + \mathbf{O}(g^2). \quad (2.25)$$

Substituting Eqs. (2.23) and (2.25) into Eq. (2.22) generates

$$S_k(\mathbf{z}_k + g\dot{\mathbf{z}}_k + \mathbf{O}(g^2)) = \mathbf{q}_k - \mathbf{q}_{k-1} + S_{k-1}(\mathbf{z}_{k-1} + g\dot{\mathbf{z}}_{k-1} + \mathbf{O}(g^2)) - \Delta\boldsymbol{\zeta}_k - \Lambda \sum_{i=0}^k \boldsymbol{\zeta}_i. \quad (2.26)$$

The above equation can be simplified as

$$S_k\mathbf{z}_k + gS_k\dot{\mathbf{z}}_k + \mathbf{O}(g^2) - \mathbf{q}_k = S_{k-1}\mathbf{z}_{k-1} + gS_{k-1}\dot{\mathbf{z}}_{k-1} - \mathbf{q}_{k-1} - \Delta\boldsymbol{\zeta}_k - \Lambda \sum_{i=0}^k \boldsymbol{\zeta}_i,$$

which can be further reformulated as

$$\boldsymbol{\zeta}_k + gS_k\dot{\mathbf{z}}_k + \mathbf{O}(g^2) = \boldsymbol{\zeta}_{k-1} + gS_{k-1}\dot{\mathbf{z}}_{k-1} - \Delta\boldsymbol{\zeta}_k - \Lambda \sum_{i=0}^k \boldsymbol{\zeta}_i. \quad (2.27)$$

Additionally, using Taylor expansion to S_k obtains

$$S_k = S_{k-1} + g\dot{S}_{k-1} + \chi(g^2), \quad (2.28)$$

where $\chi(g^2)$ represents a matrix composed of $O(g^2)$. Substituting Eq. (2.28) into Eq. (2.27) for simplification, we get

$$\begin{aligned} \boldsymbol{\zeta}_k + g(S_{k-1} + g\dot{S}_{k-1} + \chi(g^2))\dot{\mathbf{z}}_k + \mathbf{O}(g^2) = \\ \boldsymbol{\zeta}_{k-1} + gS_{k-1}\dot{\mathbf{z}}_{k-1} - \Delta\boldsymbol{\zeta}_k - \Lambda \sum_{i=0}^k \boldsymbol{\zeta}_i, \end{aligned} \quad (2.29)$$

which can be rewritten as

$$(1 + \Delta + \Lambda)\boldsymbol{\zeta}_k - \boldsymbol{\zeta}_{k-1} + \Lambda \sum_{i=0}^{k-1} \boldsymbol{\zeta}_i + \mathbf{O}(g^2) = 0. \quad (2.30)$$

The proof is thus completed. $\square$

2.4 Theoretical Analyses and Verification

In this section, we provide theoretical analyses to verify the global convergence and noise-tolerant properties of the designed model (2.14). Regarding the convergence performance of the constructed model, we offer the following Theorem 2.2.

Theorem 2.2 *The residual error synthesized by complex-valued DTND model (2.14) for solving complex time-dependent QP problem (2.5) globally converges to $O(g^2)$. That is,*

$$\lim_{k \rightarrow \infty} \|\xi_k\|_2 = O(g^2).$$

Proof Equation (2.30), as a simplified form of complex-valued DTND model (2.14), can be reformulated as

$$(1 + \Delta)\xi_{k+1} - \xi_k + \Delta \sum_{i=0}^{k+1} \xi_i + \mathbf{O}(g^2) = 0. \quad (2.31)$$

Taking the v th ($v \in (1, 2, \dots, m + n)$) element at time instant kg of the above formula, which is defined as ξ_k^v , yields

$$(1 + \Delta)\xi_{k+1}^v - \xi_k^v + \Delta \sum_{i=0}^{k+1} \xi_i^v + O(g^2) = 0. \quad (2.32)$$

Let the v th system at time instant kg of (2.31) subtract the v th system at time instant $(k + 1)g$ of (2.31); then

$$(1 + \Delta + \Lambda)\xi_{k+1}^v = (2 + \Delta)\xi_k^v - \xi_{k-1}^v + O(g^2). \quad (2.33)$$

Let

$$\eta_{k+1}^v = [\xi_{k+1}^v, \xi_k^v]^H. \quad (2.34)$$

Then, Eq. (2.33) can be rewritten as

$$\eta_{k+1}^v = A \cdot \eta_k^v + \mathbf{O}(g^2), \quad (2.35)$$

with the definition of the matrix A being

$$A = \begin{bmatrix} (2 + \Delta)/(1 + \Delta + \Lambda) & -1/(1 + \Delta + \Lambda) \\ 1 & 0 \end{bmatrix}.$$

The eigenvalues of the matrix A are recorded as α_1 and α_2 , which are $\alpha_1 = (\Delta + 2 - \sqrt{\Delta^2 - 4\Delta})/(2\Delta + 2\Lambda + 2)$, $\alpha_2 = (\Delta + 2 + \sqrt{\Delta^2 - 4\Delta})/(2\Delta + 2\Lambda + 2)$. Expanding Eq. (2.35) leads to

$$\begin{aligned}
\|\eta_{k+1}^v\|_2 &\leq \|A \cdot \eta_k^v\|_2 + \|\mathbf{O}(g^2)\|_2 \\
&= \|A \cdot \eta_k^v\|_2 + O(g^2) \\
&\leq \|A\|_2 \cdot \|\eta_k^v\|_2 + O(g^2) \\
&= \|A\|_2 \cdot \|A \cdot \eta_{k-1}^v + \mathbf{O}(g^2)\|_2 + O(g^2) \\
&\leq \|A\|_2^2 \cdot \|\eta_{k-1}^v\|_2 + O(g^2) \\
&= \dots \\
&\leq \|A\|_2^k \cdot \|\eta_1^v\|_2 + O(g^2).
\end{aligned} \tag{2.36}$$

Given that $|\text{Re}(\alpha_1)| < 1$ and $|\text{Re}(\alpha_2)| < 1$, we can draw the conclusion that $\lim_{k \rightarrow \infty} \|A\|_2^k = 0$. Therefore,

$$\lim_{k \rightarrow \infty} \|\eta_{k+1}^v\|_2 = \lim_{k \rightarrow \infty} \|A\|_2^k \cdot \|\eta_1^v\|_2 + O(g^2) = O(g^2).$$

We can get $\lim_{k \rightarrow \infty} \|\zeta_k\|_2 = O(g^2)$ by the definition of Eq. (2.34). The proof is thus completed. $\square$

In what follows, we offer theorems regarding different types of noises.

Theorem 2.3 *Considering that the presented complex-valued DTND model (2.14) is perturbed by the linear time-dependent noise $\varphi(t) = \sigma t + \rho$, its residual error for solving complex time-dependent QP problem (2.5) globally converges to $\|\frac{\sigma}{\lambda}\|_2 + O(g^2)$.*

Proof It can be generalized from Theorem 2.1 that the DTND model (2.14) is a linear system, and φ_k ($\varphi_k = \sigma k g + \rho$) and $O(g^2)$ can be analyzed separately according to the superposition of linear system. In this sense, on the basis of Eq. (2.30), the relationship between residual error and the noise component can be expressed as

$$(1 + \Delta + \Lambda)\zeta_k = \zeta_{k-1} - \Lambda \sum_{i=0}^{k-1} \zeta_i + \sigma k g + \rho, \tag{2.37}$$

which can be further written as

$$(1 + \Delta + \Lambda)\zeta_{k+1} = \zeta_k - \Lambda \sum_{i=0}^k \zeta_i + \sigma(k+1)g + \rho. \tag{2.38}$$

The v th subsystem can be formulated as

$$(1 + \Delta + \Lambda)\xi_{k+1}^v = \xi_k^v - \Lambda \sum_{i=0}^k \xi_i^v + \sigma^v(k+1)g + \rho^v. \quad (2.39)$$

Manipulating the above equation with Z-transform leads to

$$(1 + \Delta + \Lambda)(z\xi^v(z) - z\xi^v(0)) = \xi^v(z) - \Lambda \cdot \frac{\xi^v(z)}{1 - z^{-1}} + \frac{z\sigma^vg}{(z-1)^2} + \frac{z(\rho^v + \sigma^vg)}{z-1}, \quad (2.40)$$

where $\xi^v(0)$ signifies the initial value of $\xi^v(z)$. Reformulating (2.40) leads to

$$\xi^v(z) = \frac{z(z-1)^2(1 + \Delta + \Lambda)\xi^v(0) + z(z-1)(\rho^v + \sigma^vg) + z\sigma^vg}{z(z-1)^2(1 + \Delta + \Lambda) - (z-1)^2 + \Lambda z(z-1)}. \quad (2.41)$$

The poles of the above equation are $z_{1,2} = (\Delta + 2 \pm \sqrt{\Delta^2 - 4\Delta})/(2\Delta + 2\Lambda + 2)$, $z_3 = 1$. It is apparent that the absolute values of these poles are not more than 1, which means the poles are in a unit circle; thus, Eq. (2.39) is stable. So we have

$$\begin{aligned} & \lim_{k \rightarrow \infty} \xi^v(k) \\ &= \lim_{z \rightarrow 1} (z-1) \cdot \xi^v(z) \\ &= \lim_{z \rightarrow 1} \frac{(z-1)^2(1 + \Delta + \Lambda)\xi^v(0) + (z-1)(\rho^v + \sigma^vg) + \sigma^vg}{(z-1)(1 + \Delta + \Lambda) - (1 - z^{-1}) + \Lambda} \\ &= \frac{\sigma^vg}{\Lambda} = \frac{\sigma^v}{\lambda}. \end{aligned} \quad (2.42)$$

By Theorem 2.1, it can be further concluded that $\lim_{k \rightarrow \infty} \|\xi_k\|_2 = \|\frac{\sigma}{\lambda}\|_2 + O(g^2)$, regardless of how large $\varphi(t)$ is. The proof is thus completed. In addition, for any constant noises, σ is always 0 and $\lim_{k \rightarrow \infty} \xi^v(k) = 0$, which illustrates that the steady-state maximal error $\lim_{k \rightarrow \infty} \|\xi_k\|_2$ converges to $O(g^2)$. $\square$

Theorem 2.4 *Considering that the given complex-valued DTND model (2.14) is perturbed by the bounded random noise $\omega(t)$, its residual error for solving complex time-dependent QP problem (2.5) is globally convergent to*

$$2(m+n) \cdot \sup_{1 \leq i \leq k, 1 \leq v \leq (m+n)} |\omega_i^v| / (1 - \|A\|_2) + O(g^2). \quad (2.43)$$

Proof Similar to the proof in Theorem 2.3, the analysis can be separated into two parts relating to the bounded random noise $\omega(t)$ and $O(g^2)$. Based on Eq. (2.33), we can similarly obtain

$$(1 + \Delta + \Lambda)\xi_{k+1}^v = (2 + \Delta)\xi_k^v - \xi_{k-1}^v + \omega_k^v - \omega_{k-1}^v. \quad (2.44)$$

Let $u_k^v = [\omega_k^v - \omega_{k-1}^v, 0]^T$. Equation (2.44) can be reformulated into a state-space equation in control aided with the proof in Theorem 2.2 as follows:

$$\eta_{k+1}^v = A \cdot \eta_k^v + u_k^v. \quad (2.45)$$

Then, we can obtain

$$\begin{aligned} \|\eta_{k+1}^v\|_2 &\leq \|A \cdot \eta_k^v\|_2 + \|u_k^v\|_2 \\ &\leq \|A\|_2 \cdot \|\eta_k^v\|_2 + \|u_k^v\|_2 \\ &\leq \|A\|_2 \|A \cdot \eta_{k-1}^v\|_2 + \|A\|_2 \|u_{k-1}^v\|_2 + \|u_k^v\|_2 \\ &\leq \|A\|_2^2 \|\eta_{k-1}^v\|_2 + \|A\|_2 \|u_{k-1}^v\|_2 + \|u_k^v\|_2 \\ &\dots \\ &\leq \|A\|_2^k \|\eta_1^v\|_2 + \|A\|_2^{k-1} \|u_1^v\|_2 + \dots + \|u_k^v\|_2 \\ &< \|A\|_2^k \|\eta_1^v\|_2 + \max_{1 \leq i \leq k} \|u_i^v\|_2 / (1 - \|A\|_2) \\ &< \|A\|_2^k \|\eta_1^v\|_2 + 2 \max_{1 \leq i \leq k} |\omega_i^v| / (1 - \|A\|_2). \end{aligned} \quad (2.46)$$

Given that $\lim_{k \rightarrow \infty} \|A\|_2^k = 0$, the above formula can be further rearranged into

$$\lim_{k \rightarrow \infty} \|\eta_{k+1}^v\|_2 < 2 \max_{1 \leq i \leq k} |\omega_i^v| / (1 - \|A\|_2).$$

Then we can draw a conclusion that is

$$\lim_{k \rightarrow \infty} \|\zeta_k\|_2 < 2(m+n) \sup_{1 \leq i \leq k, 1 \leq v \leq (m+n)} \frac{\omega_i^v}{1 - \|A\|_2} + O(g^2).$$

The proof is thus completed. $\square$

In addition, regarding the performance of the complex-valued DTND-BFGS model (2.19), we offer the following theorems.

Theorem 2.5 Suppose that, during the sampling interval g , the approximate inverse $\tilde{U}_k$ in complex-valued DTND-BFGS model (2.19) converges to $S_k^\dagger$ with residual error $\chi(g^2)$, i.e., $\tilde{U}_k - S_k^\dagger = \chi(g^2)$, and complex-valued DTND-BFGS model (2.19) completes one updating. Then the residual error synthesized by complex-valued DTND-BFGS model (2.19) for solving complex time-dependent QP problem (2.5) globally converges to $O(g^2)$. That is, $\lim_{k \rightarrow \infty} \|\zeta_k\|_2 = O(g^2)$.

Proof During each sampling interval g , we have

$$\tilde{U}_k = S_k^\dagger + \chi(g^2). \quad (2.47)$$

Complex-valued DTND-BFGS model (2.19) can be rewritten as

$$\begin{aligned} \mathbf{z}_{k+1} = & \mathbf{z}_k + (S_k^\dagger + \chi(g^2))(-(S_k - S_{k-1})\mathbf{z}_k + \mathbf{q}_k - \mathbf{q}_{k-1} \\ & - \gamma g(S_k \mathbf{z}_k - \mathbf{q}_k) - \lambda g \sum_{i=0}^k (S_i \mathbf{z}_i - \mathbf{q}_i)), \end{aligned} \quad (2.48)$$

which can be further reformulated as

$$\begin{aligned} \mathbf{z}_{k+1} = & \mathbf{z}_k + S_k^\dagger (-(S_k - S_{k-1})\mathbf{z}_k + \mathbf{q}_k - \mathbf{q}_{k-1} \\ & - \gamma g(S_k \mathbf{z}_k - \mathbf{q}_k) - \lambda g \sum_{i=0}^k (S_i \mathbf{z}_i - \mathbf{q}_i)) + \mathbf{O}(g^2). \end{aligned} \quad (2.49)$$

The above equation can be converted into the following one with the aid of Theorem 2.1:

$$(1 + \Delta + \Lambda)\boldsymbol{\zeta}_k - \boldsymbol{\zeta}_{k-1} + \Lambda \sum_{i=0}^{k-1} \boldsymbol{\zeta}_i + \mathbf{O}(g^2) + \mathbf{O}(g^2) = 0,$$

which can be further simplified as

$$(1 + \Delta + \Lambda)\boldsymbol{\zeta}_k - \boldsymbol{\zeta}_{k-1} + \Lambda \sum_{i=0}^{k-1} \boldsymbol{\zeta}_i + \mathbf{O}(g^2) = 0.$$

Then, it can be generalized from Theorem 2.2 that the residual error synthesized by complex-valued DTND-BFGS model (2.19) for solving complex time-dependent QP problem (2.5) globally converges to $O(g^2)$. That is, $\lim_{k \rightarrow \infty} \|\boldsymbol{\zeta}_k\|_2 = O(g^2)$. Thus, the proof is completed. $\square$

Theorem 2.6 Suppose that, during the sampling interval g , the approximate inverse $\tilde{U}_k$ in complex-valued DTND-BFGS model (2.19) converges to $S_k^\dagger$ with residual error $\chi(g^2)$, i.e., $\tilde{U}_k - S_k^\dagger = \chi(g^2)$, and complex-valued DTND-BFGS model (2.19) completes one updating. Then the residual error synthesized by complex-valued DTND-BFGS model (2.19) and perturbed by the linear time-dependent noise $\varphi(t) = \sigma t + \rho$ for solving complex time-dependent QP problem (2.5) globally converges to $\left\| \frac{\sigma}{\lambda} \right\|_2 + O(g^2)$.

Proof It can be generalized from Theorem 2.3 and thus omitted. $\square$

Theorem 2.7 Suppose that, during the sampling interval g , the approximate inverse $\tilde{U}_k$ in complex-valued DTND-BFGS model (2.19) converges to $S_k^\dagger$ with residual error $\chi(g^2)$, i.e., $\tilde{U}_k - S_k^\dagger = \chi(g^2)$, and complex-valued DTND-BFGS model (2.19) completes one updating. Then the residual error synthesized by

complex-valued DTND-BFGS model (2.19) and perturbed by the bounded random noise $\omega(t)$ for solving complex time-dependent QP problem (2.5) globally converges to $2(m+n) \cdot \sup_{1 \leq i \leq k, 1 \leq v \leq (m+n)} |\omega_i^v| / (1 - \|A\|_2) + O(g^2)$.

Proof It can be generalized from Theorem 2.3 and thus omitted. $\square$

Theorem 2.8 The residual error $\|\xi_k\|_2$ of a traditional method (which does not exploit the velocity compensation in tracking the theoretical time-dependent solution) used to solve complex time-dependent QP problem (2.5) approaches $O(g)$ globally.

Proof Considering the time derivation, we have $d\mathbf{z}_k^v/dt = \lim_{g \rightarrow 0} \Delta\mathbf{z}_k^v/g = \dot{\mathbf{z}}_k^v$, where the $\mathbf{z}_k^v$ represents the v th value of $\mathbf{z}$ at the time instant $t = kg$, and the $\dot{\mathbf{z}}_k^v$ denotes the derivation of $\mathbf{z}_k^v$. We can obtain $\Delta\mathbf{z}_k^v \approx g\dot{\mathbf{z}}_k^v$. For a traditional method (which does not exploit the velocity compensation in tracking the theoretical time-dependent solution), we have $\Delta\mathbf{z}_k^v = O(g)$ and $\Delta\mathbf{z}_k = \mathbf{O}(g)$. In addition, let $\mathbf{z}_k^*$ and $\mathbf{z}_{k-1}^*$ denote the theoretical solution at $t = kg$ and $t = (k-1)g$, respectively. $\mathbf{z}_k^* = \mathbf{z}_{k-1}^* + \Delta\mathbf{z}_{k-1}$ can be obtained, which can be further rewritten as $\mathbf{z}_k^* = \mathbf{z}_{k-1}^* + \mathbf{O}(g) = \mathbf{z}_k + \mathbf{O}(g)$. Thus,

$$\begin{aligned} \|\xi_k\|_2 &= \|S_k \mathbf{z}_k - \mathbf{q}_k\|_2 \\ &= \|S_k \mathbf{z}_k^* - \mathbf{q}_k + S_k \mathbf{O}(g)\|_2 \\ &= \|S_k \mathbf{O}(g)\|_2 \\ &= \|\mathbf{O}(g)\|_2 \\ &= O(g). \end{aligned}$$

The proof is thus completed. $\square$

2.5 Numerical Experiments

To further verify the efficacy and superiority of the designed two models, an illustrative example is given with comparative results conducted.

In this section, we consider the complex time-dependent QP problem (2.5) with the following coefficients:

$$S(t) = \begin{bmatrix} 5.650 & -1.141 & -0.399 & -2.824 & -0.478 & -0.229 & -\text{isin}(5t) \\ -1.141 & 4.739 & -0.688 & 2.128 & -1.996 & 1.616 & -\text{icos}(5t) \\ -0.399 & -0.688 & 3.305 & 0.607 & 0.007 & -0.816 & -\text{isin}(5t) \\ -2.824 & 2.128 & 0.607 & 9.025 & -5.792 & -1.693 & -\text{icos}(5t) \\ -0.478 & -1.996 & 0.007 & -5.792 & 8.662 & 1.744 & -\text{isin}(5t) \\ -0.229 & 1.616 & -0.816 & -1.693 & 1.744 & 1.556 & -\text{icos}(5t) \\ \text{isin}(5t) & \text{icos}(5t) & \text{isin}(5t) & \text{icos}(5t) & \text{isin}(5t) & \text{icos}(5t) & 0 \end{bmatrix},$$

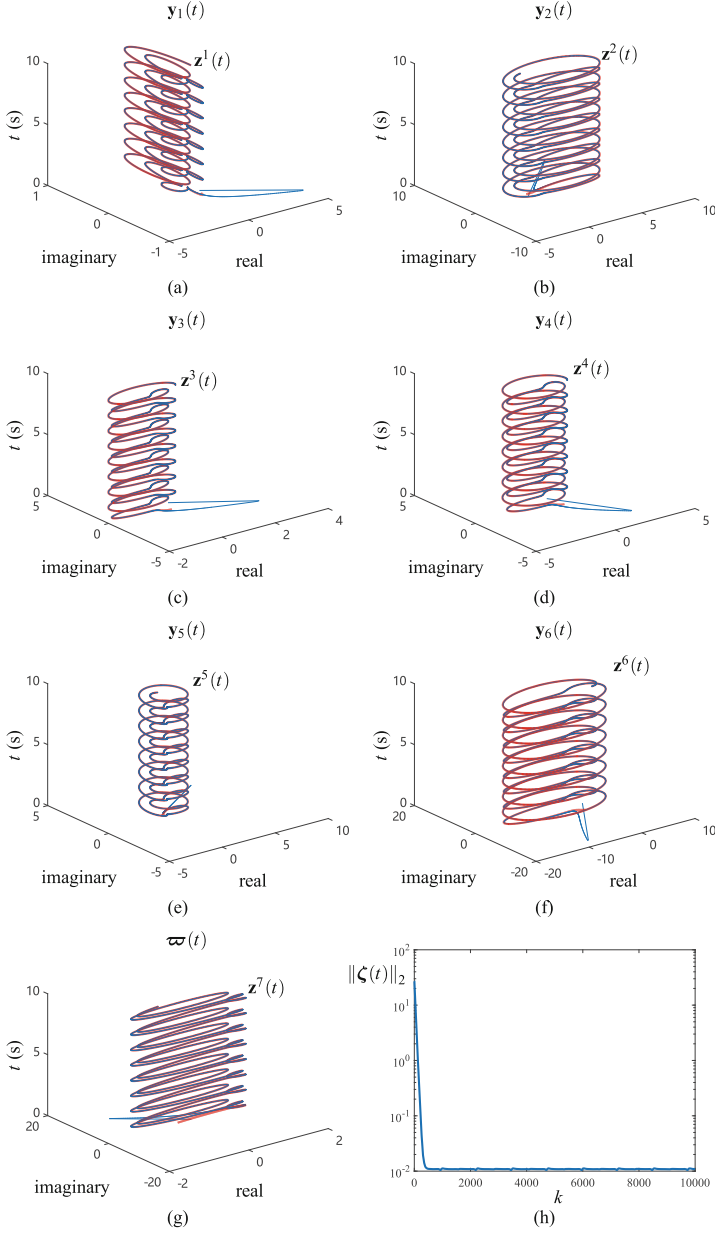


Fig. 2.1 Transient behavior of $\mathbf{z}(t)$ and residual error synthesized by complex-valued DTND model (2.14) for solving complex time-dependent QP problem (2.50) perturbed by linear time-dependent noise $\varphi(t) = \beta(100 - t)$ with $g = 0.001$ s, $\Delta = 1$, $\Lambda = 25$, and randomly generated initial values $\mathbf{z}_0 \in [0, 1]^7$ and $\beta \in [0, 1]$, where red dash-dotted curves correspond to the theoretical solutions and the blue solid curves correspond to the generated solutions. (a) Profiles of $\mathbf{z}^1(t)$ (i.e., $\mathbf{y}_1(t)$), (b) profiles of $\mathbf{z}^2(t)$ (i.e., $\mathbf{y}_2(t)$), (c) profiles of $\mathbf{z}^3(t)$ (i.e., $\mathbf{y}_3(t)$), (d) profiles of $\mathbf{z}^4(t)$ (i.e., $\mathbf{y}_4(t)$), (e) profiles of $\mathbf{z}^5(t)$ (i.e., $\mathbf{y}_5(t)$), (f) profiles of $\mathbf{z}^6(t)$ (i.e., $\mathbf{y}_6(t)$), (g) profiles of $\mathbf{z}^7(t)$ (i.e., $\mathbf{w}(t)$), and (h) profiles of residual error

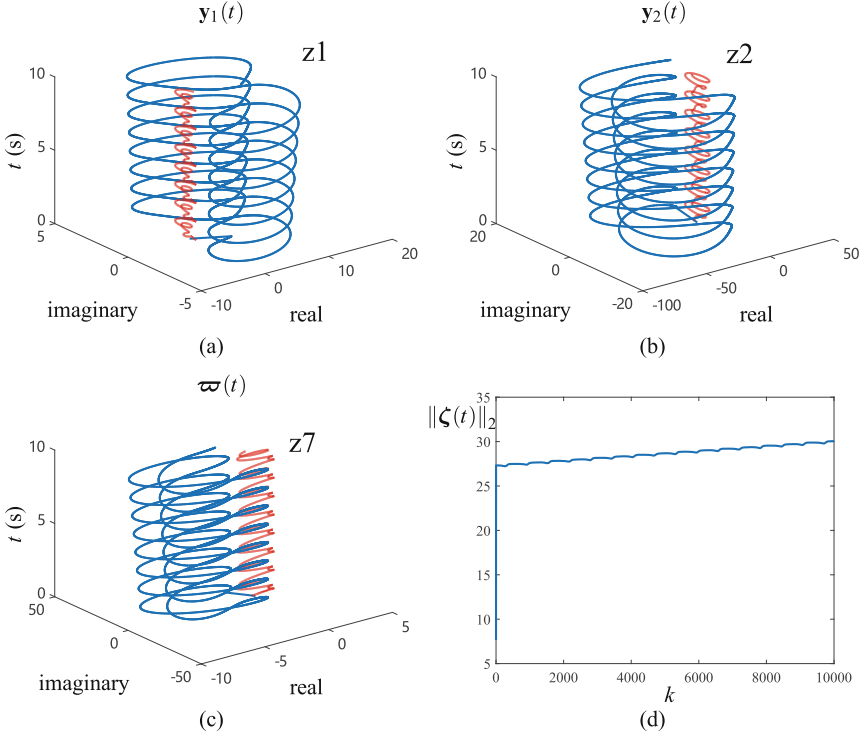


Fig. 2.2 Transient behavior of selected $\mathbf{z}(t)$ and residual error synthesized by the complex-valued Newton iteration (2.17) for solving complex time-dependent QP problem (2.50) perturbed by linear time-dependent noise $\varphi(t) = \beta(100 - t)$ with $g = 0.001$ s and randomly generated initial values $\mathbf{z}_0 \in [0, 1]^7$ and β defined the same as that in Fig. 2.1, where red dash-dotted curves correspond to the theoretical solutions and the blue solid curves correspond to the generated solutions. (a) Profiles of $\mathbf{z}^1(t)$ (i.e., $\mathbf{y}_1(t)$), (b) profiles of $\mathbf{z}^2(t)$ (i.e., $\mathbf{y}_2(t)$), (c) profiles of $\mathbf{z}^7(t)$ (i.e., $\mathbf{w}(t)$), and (d) profiles of residual error

$$\mathbf{q}(t) = [\exp(-5it) \quad \exp(-10it) \quad \exp(-5it) \quad \exp(-10it) \quad \exp(-5it) \quad \exp(-10it) \quad \exp(-10it) + 1]^H. \quad (2.50)$$

The corresponding numerical experimental results are shown in Figs. 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, and 2.7 as well as Table 2.1 with comparisons with Newton iteration (2.17), traditional DTZNN model (2.16), and CCGP algorithm (2.18).

Specifically, Fig. 2.1 visualizes the numerical experimental results synthesized by the given complex-valued DTND model (2.14) for solving complex time-dependent QP problem (2.50) perturbed by linear time-dependent noise $\varphi(t) = \beta(100 - t)$ with $g = 0.001$ s, $\Delta = 1$, $\Lambda = 25$, and randomly generated initial values $\mathbf{z}_0 \in [0, 1]^7$ and $\beta \in [0, 1]$. It can be observed from the figure that the generated solutions (denoted by blue solid curves) rapidly converge to the theoretical ones (denoted by red dash-

Fig. 2.3 Convergence performance of the complex-valued DTND model (2.14) (denoted by CD), traditional DTZNN model (2.16) (denoted by TD), CCGP algorithm (2.18) (denoted by CC), and Newton iteration (2.17) (denoted by NI) for solving the perturbed complex time-dependent QP problem (2.50) without perturbations. **(a)** $g = 0.0001$ s, **(b)** $g = 0.001$ s, and **(c)** $g = 0.01$ s

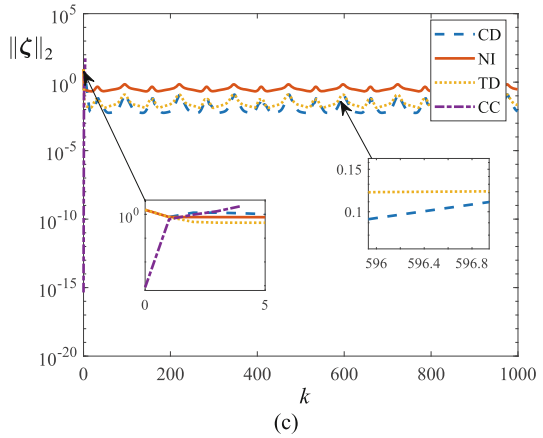
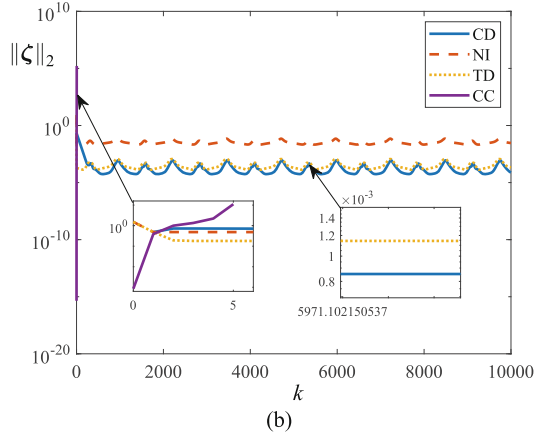
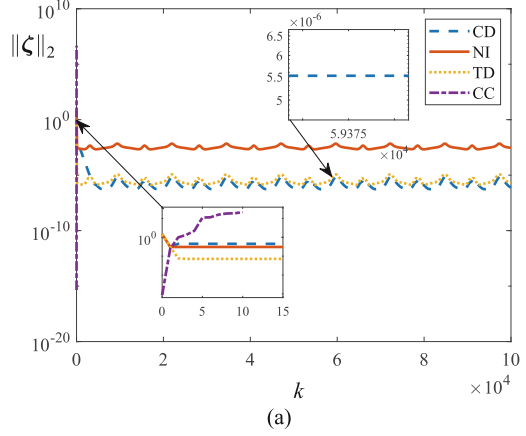


Fig. 2.4 Convergence performance of the presented complex-valued DTND model (2.14) (denoted by CD), traditional DTZNN model (2.16) (denoted by TD), CCGP algorithm (2.18) (denoted by CC), and Newton iteration (2.17) (denoted by NI) for solving the perturbed complex time-dependent QP problem (2.50) polluted with randomly generated constant noises. (a) $g = 0.0001$ s, (b) $g = 0.001$ s, and (c) $g = 0.01$ s

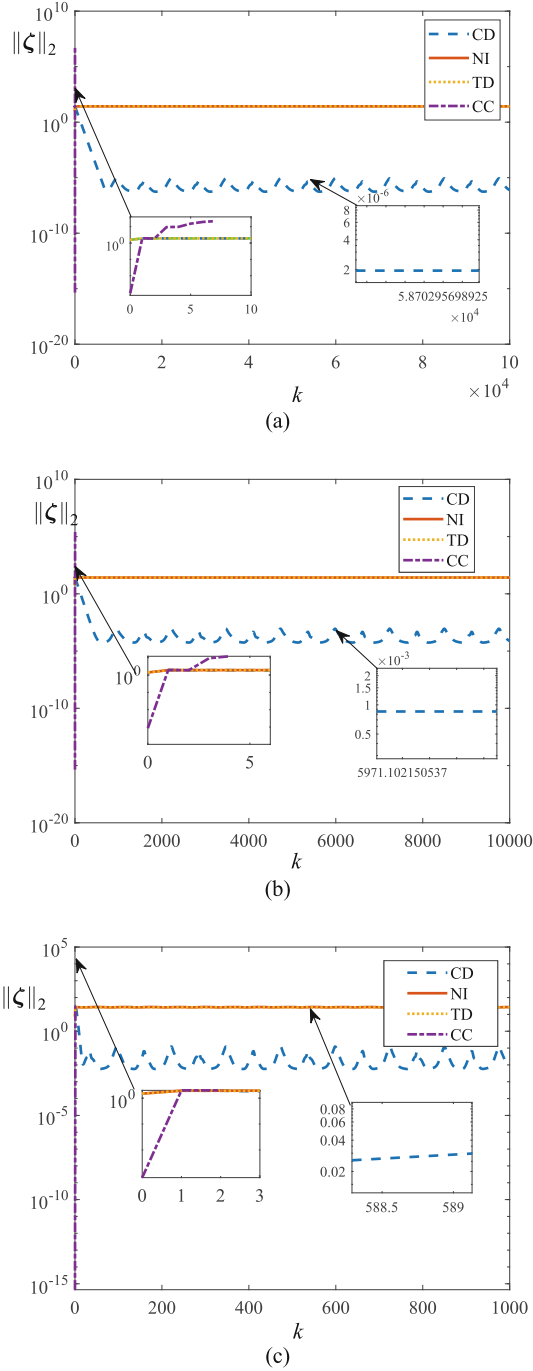


Fig. 2.5 Convergence performance of the designed complex-valued DTND model (2.14) (denoted by CD), traditional DTZNN model (2.16) (denoted by TD), CCGP algorithm (2.18) (denoted by CC), and Newton iteration (2.17) (denoted by NI) for solving the perturbed complex time-dependent QP problem (2.50) disturbed by linear time-dependent noise $\varphi(t) = 10 * (t - 1)$. **(a)** $g = 0.0001$ s, **(b)** $g = 0.001$ s, and **(c)** $g = 0.01$ s

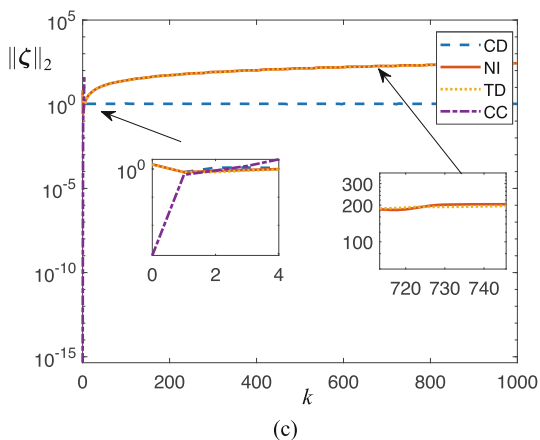
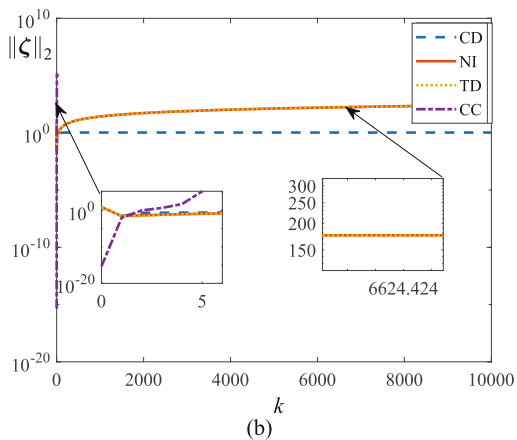
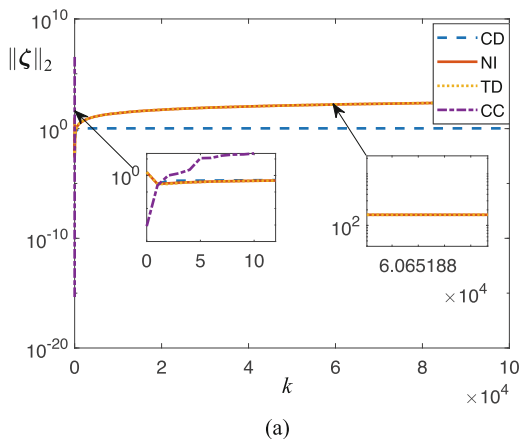


Fig. 2.6 Convergence performance of the designed complex-valued DTND model (2.14) (denoted by CD), traditional DTZNN model (2.16) (denoted by TD), CCGP algorithm (2.18) (denoted by CC), and Newton iteration (2.17) (denoted by NI) for solving the perturbed complex time-dependent QP problem (2.50) with random noise

$\varphi(t) \in [-1, 1] * 0.5 + 100$.

(a) $g = 0.0001$ s, (b)

$g = 0.001$ s, and (c)

$g = 0.01$ s

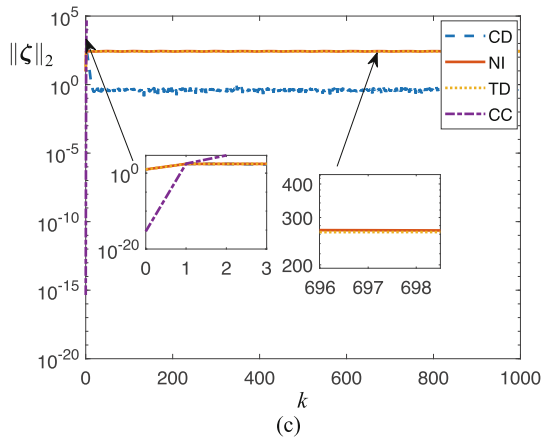
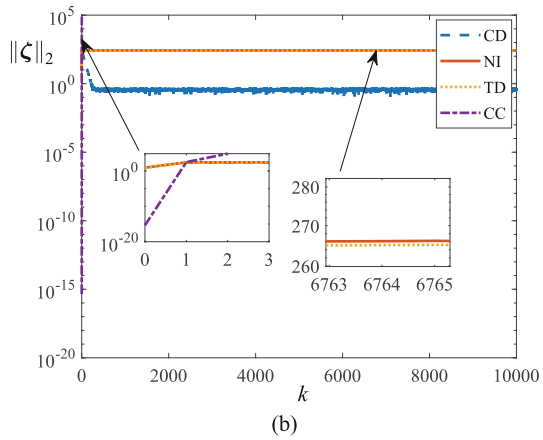
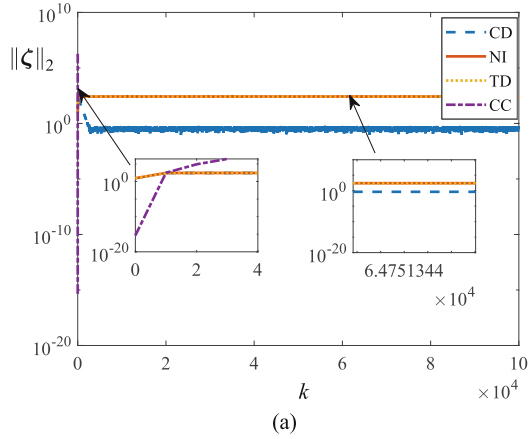
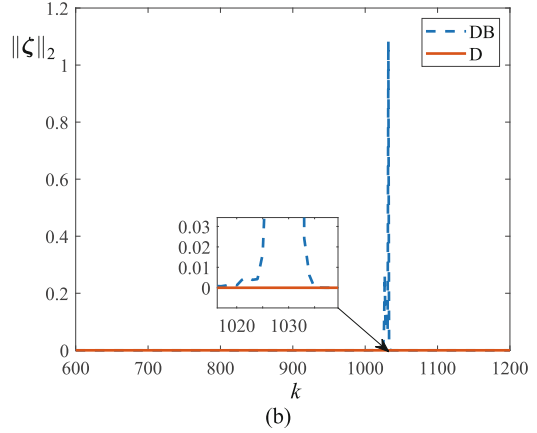
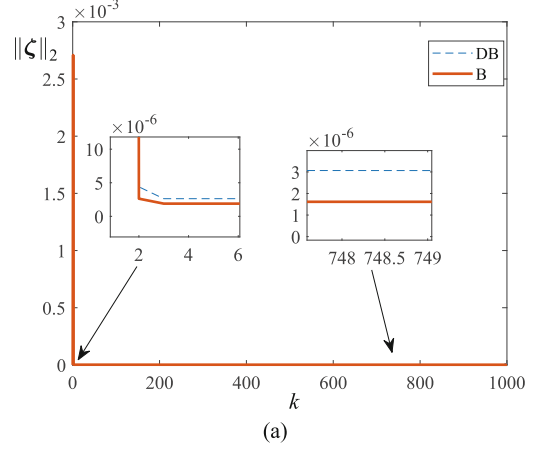


Fig. 2.7 Comparison of complex-valued DTND model (2.14) (denoted by DTND) and complex-valued DTND-BFGS model (2.19) (denoted by BFGS) for solving complex time-dependent QP problem (2.50) with $g = 0.0001$ s. (a) Convergence performance of the two models and (b) sudden protuberance of the residual error of complex-valued DTND-BFGS model (2.19)



dotted curves) with the residual error shown in Fig. 2.1h and of the order 10^{-2} . The results illustrated in Fig. 2.1 preliminarily validate the correctness of the proof in Theorem 2.3.

In addition, for comparison, Newton iteration (2.17) is employed to solve complex time-dependent QP problem (2.50) with the same perturbation, of which the numerical experimental results are shown in Fig. 2.2. It is clearly shown in the figure that the computed solutions via Newton iteration (2.17) (denoted by blue solid curves) cannot converge to the theoretical ones (denoted by red dash-dotted curves) with the residual error shown in Fig. 2.2d, which increases linearly with time. In this sense, the residual error of Newton iteration (2.17) for solving complex time-dependent QP problem (2.50) is of infinity.

Figure 2.3 shows the numerical experimental comparison results of Newton iteration (2.17), traditional DTZNN model (2.16), CCGP algorithm (2.18), and the constructed DTND model (2.14) without perturbations. It can be seen that

Newton iteration (2.17) and CCGP algorithm (2.18) have poor performance, which, evidently, is due to the lack of velocity compensation of the dynamic parameters in achieving an accuracy solution. In contrast, by exploiting the time-derivative information directly or by using the differential information of the parameters, traditional DTZNN model (2.16) or DTND model (2.14) is able to track the time-dependent exact solution with enough accuracy. Specifically, as shown in Table 2.1 as well as Fig. 2.3a, for $g = 0.0001$ s with the average computing time per updating (ACTPU) less than the sampling gap g , the maximal steady-state residual errors (MSSREs) synthesized by traditional DTZNN model (2.16), Newton iteration

Table 2.1 Comparisons among different models (complex-valued DTND model (2.14), Newton iteration (2.17), traditional DTZNN model (2.16), and CCGP algorithm (2.18)) for solving complex time-dependent QP problem (2.50) in terms of MSSRE with the average computing time per updating (ACTPU) in brackets, where their initial states are generated randomly

g (s)	Model	MSSRE and ACTPU (s) in brackets without noises	MSSRE and ACTPU (s) in brackets with randomly generated constant noises	MSSRE and ACTPU (s) in brackets with linear noises $\varphi(t) = 10 * (t - 1)$	MSSRE and ACTPU (s) in brackets with random noises $\varphi(t) \in [99.5, 100.5]$
0.01	Model (2.14)	6.740×10^{-2} (3.679×10^{-5})	6.740×10^{-2} (3.670×10^{-5})	1.098×10^0 (3.593×10^{-5})	6.839×10^{-1} (3.637×10^{-5})
	Model (2.17)	7.012×10^{-1} (1.140×10^{-5})	2.763×10^1 (1.128×10^{-5})	Infinity	2.780×10^2 (1.165×10^{-5})
	Model (2.16)	1.216×10^{-1} (2.212×10^{-5})	2.677×10^1 (2.183×10^{-5})	Infinity	2.687×10^2 (2.213×10^{-5})
	Model (2.18)	5.476×10^1 (5.288×10^{-5})	NA	NA	NA
0.001	Model (2.14)	8.918×10^{-4} (3.014×10^{-5})	8.918×10^{-4} (3.002×10^{-5})	1.059×10^0 (3.096×10^{-5})	5.696×10^{-1} (3.150×10^{-5})
	Model (2.17)	7.020×10^{-2} (8.803×10^{-6})	2.656×10^1 (8.906×10^{-6})	Infinity	2.668×10^2 (9.698×10^{-6})
	Model (2.16)	1.100×10^{-3} (1.730×10^{-5})	2.646×10^1 (1.726×10^{-5})	Infinity	2.657×10^2 (1.841×10^{-5})
	Model (2.18)	NA	NA	NA	NA
0.0001	Model (2.14)	8.731×10^{-6} (2.920×10^{-5})	8.731×10^{-6} (2.918×10^{-5})	1.058×10^0 (3.023×10^{-5})	6.013×10^{-1} (3.076×10^{-5})
	Model (2.17)	7.000×10^{-3} (8.678×10^{-6})	2.647×10^1 (8.523×10^{-6})	Infinity	2.658×10^2 (9.537×10^{-6})
	Model (2.16)	1.144×10^{-5} (1.687×10^{-5})	2.646×10^1 (1.682×10^{-5})	Infinity	2.658×10^2 (1.801×10^{-5})
	Model (2.18)	NA	NA	NA	NA

“NA” means that CCGP algorithm (2.18) is not able to output $\mathbf{y}(t)$ at each data point during the calculation

(2.17), and complex-valued DTND model (2.14) are of order 10^{-5} , 10^{-3} , and 10^{-6} , respectively, while CCGP algorithm (2.18) is of order 10^6 , which demonstrate the superiority of the presented model (2.14). Moreover, it can be observed that the residual error of traditional DTZNN model (2.16) or DTND model (2.14) changes in an $O(g^2)$ manner while that of Newton iteration (2.17) changes in an $O(g)$ manner. It is worth pointing out here that, even without external perturbations, any computational method may suffer from the noises due to the existing of truncation errors and roundoff errors. In this situation, as shown in Table 2.1 and Fig. 2.3, the given DTND model (2.14) performs better than the traditional DTZNN model (2.16) in view of the noise-suppressing feature. Besides, CCGP algorithm (2.18) shows the worst performance among these models and cannot complete the computation during the sampling gap, thereby leading to failure. These results further validate the correctness of the proofs in Theorems 2.2 and 2.8.

Regarding the constant noises, Fig. 2.4 and Table 2.1 present the corresponding computer simulation results. Specifically, for sampling gap $g = 0.0001$ s, Fig. 2.4a shows that the residual error of the designed DTND model (2.14) achieves rapid convergence in approaching zero with the MSSRE (listed in Table 2.1) at the order of 10^{-6} and the ACTPU being 2.918×10^{-5} . In contrast, the MSSRE of Newton iteration (2.17) or traditional DTZNN model (2.16) in the presence of randomly generated constant noises is of the order 10^1 , which is about 10^7 times larger than that of the constructed DTND model (2.14). Besides, as clearly shown in Table 2.1, the MSSRE of the given DTND model (2.14) is of $O(g^2)$ manner, which verifies the theoretical analyses presented in Theorem 2.3. Besides, CCGP algorithm (2.18) is not able to output $\mathbf{y}(t)$ at each data point during the calculation and thus cannot complete the given task.

As a continuation, Figs. 2.5 and 2.6 provide results on the residual errors synthesized by these models with respect to the linear noises and random noises, respectively. It can be observed from these two figures and Table 2.1 that, for linear noises, the MSSRE of complex-valued DTND model (2.14) remains at a certain value while the rest goes to infinity and that, for nonzero mean random noises, complex-valued DTND model (2.14) performs the best among these computational methods. These results further illustrate that the presented model (2.14) has a strong stability and can remedy the weaknesses of traditional computational methods in solving complex time-dependent QP problems effectively.

As stated previously, complex-valued DTND-BFGS model (2.19) employs the BFGS method to approximate the inverse of the Hessian matrix during the computation, which requires that the error between the approximated matrix and the inverse of the Hessian matrix at each time instant should be tiny enough. Figure 2.7 plots the comparative results synthesized by these two methods. Specifically, Fig. 2.7a shows that the two methods have a similar performance in solving complex time-dependent QP problem (2.50) with $g = 0.0001$ s. However, as shown in Fig. 2.7b, a sudden protuberance is observed in the solving process at around the 1030th updating, which means that the solution generated by complex-valued DTND-BFGS model (2.19) is far away from the theoretical one. In contrast, the residual error of

complex-valued DTND model (2.14) is of the same order as the previous. Evidently, this is due to the failure in approximating the inverse of the Hessian matrix by the BFGS method. In addition, due to the superiority in handling perturbations, complex-valued DTND-BFGS model (2.19) is able to output the accurate solution after the sudden protuberance. These results further illustrate the effectiveness and superiority of the designed models.

2.6 Applications

This section presents applications of the constructed models in robotics and filter design.

2.6.1 Applications to Robotics

It is widely reported and investigated to describe and solve the kinematics control of redundant robot manipulators in the general framework of optimization [37–39], e.g., the quadratic programming is exploited in [1] to formulate the redundancy resolution of redundant robot manipulators. Therefore, the given complex-valued DTND model (2.14) is applied to the motion generation of redundant robot manipulators in this part. We specially consider the PUMA560 robot and the generalization to other redundant robot manipulators can be similarly derived and thus omitted here. The affine system at the joint-velocity level originating from the forward kinematics equation of PUMA560 robot is formulated as

$$J(\theta(t))\dot{\theta}(t) = \dot{\mathbf{p}}(t), \quad (2.51)$$

where $J(\theta(t))$ is the Jacobian matrix with $J(\theta(t)) \in \mathbb{R}^{3 \times 6}$; $\theta(t) \in \mathbb{R}^6$ denotes the joint angle vector of the PUMA560 robot; and $\mathbf{p}(t) \in \mathbb{R}^3$ is the joint position vector with $\dot{\mathbf{p}}(t)$ being its time derivative. The PUMA560 robot is expected to track a Lissajous path with the execution time of 10 s and g is set as 0.001 s. The simulation results synthesized by the presented DTND model (2.14) are shown as Figs. 2.8, 2.9, and 2.10 with three cases being taken into account: the case without perturbation, the case with constant noises, and the case with random noises, where the position error $\mathbf{e} = \mathbf{p}(t) - \mathbf{p}_d(t)$ with $\mathbf{p}_d(t)$ denoting the desired trajectory. It can be seen from these figures that the position errors generated by the designed DTND model (2.14) are all of the order 10^{-4} , which indicates the strong noise-suppressing property of the constructed DTND model (2.14).

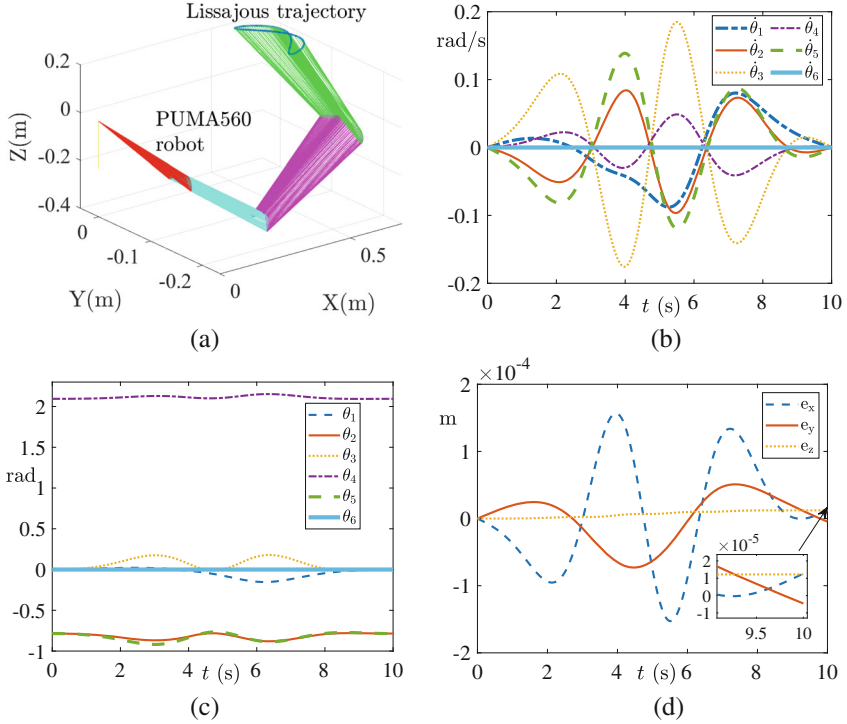


Fig. 2.8 Simulation results of PUMA560 robot tracking the Lissajous trajectory synthesized by DTND model (2.14) with $g=0.001$ s. (a) Motion trajectory (b) profiles of joint velocity, (c) profiles of joint angles, and (d) profiles of position error

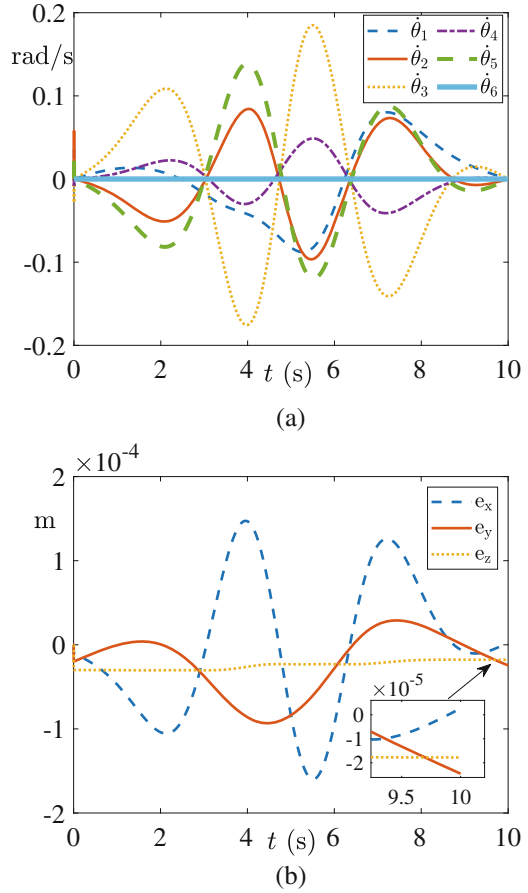
2.6.2 Applications to Complex-Valued FIR Filter Design

The finite impulse response (FIR) filters are popular in waveform transmission, image processing, and industrial production. However, research on complex-valued filters is rare compared to its wide applications. In this part, the real-valued FIR filter is extended to the complex domain by taking advantage of the given complex-valued DTND model (2.14). In the numerical experiments, the example is designed as a four-tap complex-valued FIR filter, which is expressed as

$$\hat{\vartheta}(v) = u_0 i(v) + u_1 i(v-1) + u_2 i(v-2) + u_3 i(v-3), \quad (2.52)$$

where $\vartheta(v)$ is the output signal of the filter; v denotes the v th input or output; $i(v)$ signifies the input signal sequence; and u_0, u_1, u_2, u_3 denote tap weights of the filter. The equation provides an optimal signal estimation with minimum sum of squared output errors. Furthermore, the above formula can be rewritten as

Fig. 2.9 Simulation results of PUMA560 robot tracking the Lissajous trajectory synthesized by DTND model (2.14) with constant noise $\varphi(t) = [10; 20; 30]$ and $g=0.001$ s. **(a)** Profiles of joint velocity and **(b)** profiles of position error



$$\hat{\boldsymbol{\vartheta}} = \boldsymbol{\Omega} \mathbf{u}, \quad (2.53)$$

where

$$\boldsymbol{\Omega} = \begin{bmatrix} i(3) & i(2) & i(1) & i(0) \\ i(4) & i(3) & i(2) & i(1) \\ i(5) & i(4) & i(3) & i(2) \\ \vdots & \vdots & \vdots & \vdots \\ i(17) & i(16) & i(15) & i(14) \end{bmatrix},$$

Fig. 2.10 Simulation results of PUMA560 robot tracking the Lissajous trajectory synthesized by DTND model (2.14) with random noise $\varphi(t) \in [-1, 1] + 1$ and $g=0.001$ s. **(a)** Profiles of joint velocity and **(b)** profiles of position error

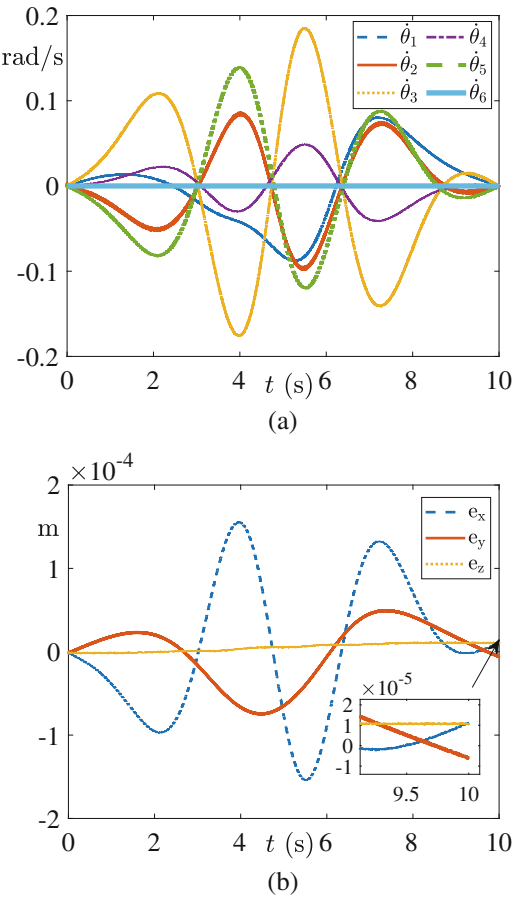


Table 2.2 The input sequence $i(v)$ and the desired output sequence $\hat{\vartheta}(v)$ of the FIR filter example

v	$i(v)$	$\hat{\vartheta}(v)$	v	$i(v)$	$\hat{\vartheta}(v)$
0	$10.5011+i9.79t$	$10.0033+i9.26t$	9	$1.2287+i1.81t$	$0.0000-i0.00t$
1	$9.8553+i8.84t$	$8.8615+i7.90t$	10	$-1.5131-i1.66t$	$-1.5643-i0.69t$
2	$8.8311+i7.56t$	$7.1002+i6.36t$	11	$-2.5173-i2.27t$	$-3.0902-i2.17t$
3	$9.5674+i8.42t$	$8.0902+i7.41t$	12	$-3.6836-i3.58t$	$-4.5399-i3.78t$
4	$6.4881+i5.21t$	$7.0711+i6.39t$	13	$-6.9138-i5.40t$	$-5.8779-i5.13t$
5	$7.5587+i6.63t$	$5.8779+i5.13t$	14	$-6.1127-i5.12t$	$-7.0711-i6.39t$
6	$4.8713+i2.34t$	$4.5399+i3.78t$	15	$-8.1881-i7.36t$	$-8.0902-i7.41t$
7	$4.5215+i2.05t$	$3.0902+i2.17t$	16	$-8.0528-i7.02t$	$-8.9101-i8.04t$
8	$0.6072+i0.79t$	$1.5643+i0.69t$	17	$-8.6237-i7.76t$	$-9.5106-i8.96t$

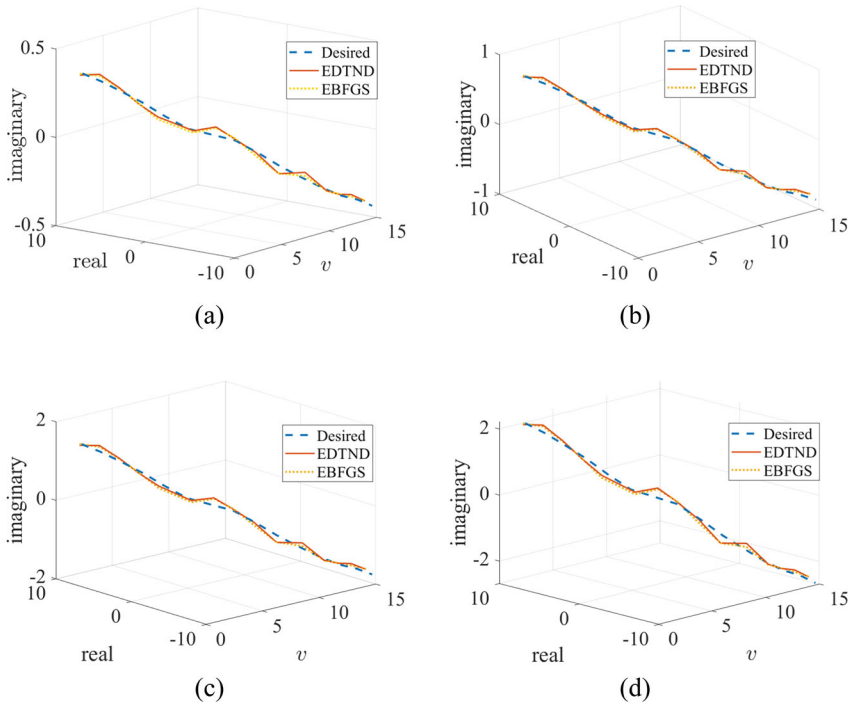


Fig. 2.11 The desired and estimated output curves of the complex-valued FIR filter (2.52) synthesized by complex-valued DTND model (2.14) (denoted by DTND) and complex-valued DTND-BFGS model (2.19) (denoted by BFGS) at different times with $g=0.001$ s. (a) $t = 0.05$ s, (b) $t = 0.1$ s, (c) $t = 0.2$ s, and (d) $t = 0.3$ s

$$\mathbf{u} = \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \end{bmatrix}, \quad \hat{\mathbf{v}} = \begin{bmatrix} \hat{v}(3) \\ \hat{v}(4) \\ \hat{v}(5) \\ \vdots \\ \hat{v}(17) \end{bmatrix}.$$

To get the least squares solution, Eq. (2.53) is converted to a quadratic programming problem with the performance index designed as the minimal norm of $\mathbf{u}$:

$$\begin{aligned} &\text{Minimize} && \frac{1}{2} \mathbf{u}^H \mathbf{u}, \\ &\text{Subject to} && \Omega \mathbf{u} = \hat{\mathbf{v}}. \end{aligned} \tag{2.54}$$

The coefficient matrix and vector of complex time-dependent QP problem (2.54) are shown in Table 2.2 [40]. Using the presented DTND model (2.14) and complex-

valued DTND-BFGS model (2.19) to solve (2.54), we obtain a set of tap weights, which are complex time-dependent. Furthermore, the estimated output signals $\hat{\mathbf{d}}$ at different time instants are obtained as shown in Fig. 2.11, which are denoted by blue solid curves (solution by DTND model (2.14)) and green dash-dotted curves (solution by DTND-BFGS model (2.19)). It is worth noting that $\Omega \mathbf{u} = \hat{\mathbf{d}}$ is an overdetermined system, and there does not exist an exact solution that has zero error. The least squares solution obtained by solving this equation is the best fitted curve available. For complex time-dependent QP problem (2.54), four solutions at different moments are selected for demonstration, and the estimated solutions (denoted by blue solid curves or green dash-dotted curves) approach the theoretical solutions (denoted by red dash curves), which illustrates that the designed DTND model (2.14) and complex-valued DTND-BFGS model (2.19) are applied to the design of complex time-dependent FIR filters successfully.

2.7 Summary

In this chapter, two complex-valued discrete-time neural dynamics models, i.e., DTND (2.14) and DTND-BFGS (2.19), have been constructed to solve time-dependent complex QP problems for the first time. Rigorous theoretical analyses have been provided to reveal the superiorities of the given models in the presence of various kinds of noises or perturbations. For comparison, Newton iteration (2.17), traditional DTZNN model (2.16), and CCGP algorithm (2.18) have been presented to solve the same problem. The superiority and effectiveness of the presented models are highlighted via the comparative computer simulative and numerical experimental results. Finally, the designed models have been applied to the motion generation and control of robotics and the design of complex-valued FIR filters, which demonstrates the efficacy and practicability of the constructed models.

References

1. Chen D, Zhang Y (2017) Robust zeroing neural-dynamics and its time-varying disturbances suppression model applied to mobile robot manipulators. *IEEE Trans Neural Netw Learn Syst* 29(9):4385–4397
2. Marashdeh Q, Warsito W, Fan LS, Teixeira FL (2006) A nonlinear image reconstruction technique for ECT using a combined neural network approach. *Meas Sci Technol* 17(8):2097–2103
3. Mattingley J, Boyd S (2010) Real-time convex optimization in signal processing. *IEEE Signal Process Mag* 27(3):50–61
4. Zhang Y, Chen S, Li S, Zhang Z (2018) Adaptive projection neural network for kinematic control of redundant manipulators with unknown physical parameters. *IEEE Trans Ind Electron* 65(6):4909–4920
5. Svoboda JA, Dorf RC (2010) Introduction to electric circuits. Wiley, New Jersey
6. Oppenheim AV, Willsky AS, Nawab SH, Ding JJ (2014) Signals and systems. Person, Harlow

7. Franklin GF, Powell JD, Emami-Naeini A (1994) Feedback control of dynamic systems. Addison-Wesley, Massachusetts
8. Liao W, Wang J (2010) A recurrent neural network for solving complex-valued quadratic programming problems with equality constraints. *Adv Swarm Intell First Int Conf* 6416:321–326
9. Saad Y, Van Der Vorst HA (2000) Iterative solution of linear systems in the 20th century. *J Comput Appl Math* 123(1–2):1–33
10. Chen X, Jin L, Hu B (2024) A cerebellum-inspired control scheme for kinematic control of redundant manipulators. *IEEE Trans Ind Electron*. In Press. <https://doi.org/10.1109/TIE.2023.3312427>
11. Liu M, Hu Y, Li J, Jin L (2024) Characteristics-capturing neural dynamics for solving time-dependent nonlinear equations with periodic noise. *IEEE Trans Cogn Dev Syst*. In Press. <https://doi.org/10.1109/TCDS.2023.3316776>
12. Jin L, Li S, Liao B, Zhang Z (2017) Zeroing neural networks: a survey. *Neurocomputing* 267:597–604
13. Guo D, Nie Z, Yan L (2017) Novel discrete-time Zhang neural network for time-varying matrix inversion. *IEEE Trans Syst Man Cybern Syst* 47(8):2301–2310
14. Li J, Zhang Y, Li S, Mao M (2017) New discretization-formula-based zeroing dynamics for real-time tracking control of serial and parallel manipulators. *IEEE Trans Ind Inform* 14(8):3416–3425
15. Jin L, Li S, La HM, Luo X (2017) Manipulability optimization of redundant manipulators using dynamic neural networks. *IEEE Trans Ind Electron* 64(6):4710–4720
16. Zhang Y, Li S, Liu X (2017) Adaptive near-optimal control of uncertain systems with application to underactuated surface vessels. *IEEE Trans Control Syst Technol* 26(4):1204–1218
17. Xiao L, Liao B, Li S, Zhang Z, Ding L, Jin L (2017) Design and analysis of FTZNN applied to the real-time solution of a nonstationary Lyapunov equation and tracking control of a wheeled mobile manipulator. *IEEE Trans Ind Inform* 14(1):98–105
18. Jin L, Li S, Hu B (2017) RNN models for dynamic matrix inversion: a control-theoretical perspective. *IEEE Trans Ind Inform* 14(1):189–199
19. Jin L, Li S, Luo X, Li Y, Qin B (2018) Neural dynamics for cooperative control of redundant robot manipulators. *IEEE Trans Ind Inform* 14(9):3812–3821
20. Jin L, Zhang Y, Qiu B (2018) Neural network-based discrete-time Z-type model of high accuracy in noisy environments for solving dynamic system of linear equations. *Neural Comput Appl* 29:1217–1232
21. Lu Y, Li D, Xu Z, Xi Y (2014) Convergence analysis and digital implementation of a discrete-time neural network for model predictive control. *IEEE Trans Ind Electron* 61(12):7035–7045
22. Liu Q, Cao J (2011) Global exponential stability of discrete-time recurrent neural network for solving quadratic programming problems subject to linear constraints. *Neurocomputing* 74(17):3494–3501
23. Li S, Wang H, Rafique MU (2017) A novel recurrent neural network for manipulator control with improved noise tolerance. *IEEE Trans Neural Netw Learn Syst* 29(5):1908–1918
24. Li J, Zhang Y, Mao M (2018) General square-pattern discretization formulas via second-order derivative elimination for zeroing neural network illustrated by future optimization. *IEEE Trans Neural Netw Learn Syst* 30(3):891–901
25. Zhang Y, Gong H, Yang M, Li J, Yang X (2018) Stepsize range and optimal value for Taylor-Zhang discretization formula applied to zeroing neurodynamics illustrated via future equality-constrained quadratic programming. *IEEE Trans Neural Netw Learn Syst* 99:1–8
26. Li S, Zhang Y, Jin L (2016) Kinematic control of redundant manipulators using neural networks. *IEEE Trans Neural Netw Learn Syst* 28(10):2243–2254
27. Qiu B, Zhang Y, Yang Z (2018) New discrete-time ZNN models for least-squares solution of dynamic linear equation system with time-varying rank-deficient coefficient. *IEEE Trans Neural Netw Learn Syst* 29(11):5767–5776

28. Zhang Y, Li Z (2009) Zhang neural network for online solution of time-varying convex quadratic program subject to time-varying linear-equality constraints. *Phys Lett A* 373(18–19):1639–1643
29. Liao B, Zhang Y, Jin L (2015) Taylor $O(h^3)$ discretization of ZNN models for dynamic equality-constrained quadratic programming with application to manipulators. *IEEE Trans Neural Netw Learn Syst* 27(2):225–237
30. Jin L, Zhang Y, Li S, Zhang Y (2016) Modified ZNN for time-varying quadratic programming with inherent tolerance to noises and its application to kinematic redundancy resolution of robot manipulators. *IEEE Trans Ind Electron* 63(11):6978–6988
31. Zhang S, Xia Y (2015) Two fast complex-valued algorithms for solving complex quadratic programming problems. *IEEE Trans Cybern* 46(12):2837–2847
32. Jin L, Zhang Y, Li S, Zhang Y (2016) Noise-tolerant ZNN models for solving time-varying zero-finding problems: a control-theoretic approach. *IEEE Trans Autom Control* 62(2):992–997
33. Ding S, Yang M, Mao M, Xiao L, Zhang Y (2016) Complex ZNN and GNN models for time-varying complex quadratic programming subject to equality constraints. In: 2016 12th world congress on intelligent control and automation, pp 210–215
34. Hu JL, Wu Z, McCann H, Davis L, Xie CG (2006) BFGS quasi-Newton method for solving electromagnetic inverse problems. *IEEE Proc-Microwaves Antennas Propag* 153(2):199–204
35. Leithead WE, Zhang Y (2007) $O(n^2)$ -operation approximation of covariance matrix inverse in Gaussian process regression based on quasi-Newton BFGS method. *Commun Stat-Simul Comput* 36(2):367–380
36. 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, pp 3711–3716
37. Li S, Zhou M, Luo X (2017) Modified primal-dual neural networks for motion control of redundant manipulators with dynamic rejection of harmonic noises. *IEEE Trans Neural Netw Learn Syst* 29(10):4791–4801
38. Li S, He J, Li Y, Rafique MU (2016) Distributed recurrent neural networks for cooperative control of manipulators: a game-theoretic perspective. *IEEE Trans Neural Netw Learn Syst* 28(2):415–426
39. Zhang Y, Li S (2018) A neural controller for image-based visual servoing of manipulators with physical constraints. *IEEE Trans Neural Netw Learn Syst* 29(11):5419–5429
40. Tang WS (2000) A discrete-time recurrent neural network for solving quadratic programs with application to FIR filter synthesis. *IEEE Int Conf Syst Man Cybern* 4:2491–2496

Chapter 3

Noise-Tolerant Neural Dynamics



3.1 Introduction

To date, due to the important role that the nonlinear optimization problem plays in various areas [1–9], many numerical methods and neural dynamics have been developed and extended to solve it, among which gradient-related methods and Newton-Raphson iteration (NRI) and their modifications are commonly used [2]. For example, a class of nonlinear conjugate gradient methods aiming at solving optimization problems are summarized in [10], which are of global convergence properties. More recently, a three-term conjugate gradient model providing descent searching directions is investigated in [11]. It is worth pointing out that a large number of practical problems are dynamic in nature [12, 13], of which the parameters involved are varying with time, thereby leading to a time-dependent theoretical solution. When solved by these traditional models, a dynamic optimization problem is assumed to be time-invariant during the computational interval and thus the generated solution is directly employed to the problem at the next time instant. This is mainly due to the fact that, without leveraging the velocity compensation for the dynamic parameters, a traditional model is not able to track the time-dependent theoretical solutions in a predictive manner [14]. Therefore, for a time-dependent problem aided with a traditional model, large lagging error is unavoidable.

Neural networks and the related neural dynamics methods [15, 16] have shown superior properties in parallel distribution [17–19] and high-speed computing [20–22] with extensive applications in neurophysiology, chemical equilibrium, and robotics [23–27]. For instance, Liu and Tong present an adaptive neural network based on optimal control for a class of nonlinear discrete-time systems in [17], which achieves optimal control performance with system stability guaranteed. Continuous-time neural dynamics is reported to be able to track the time-dependent solution of dynamic problems in an error-free manner [20]. A discrete-time

numerical algorithm based on neural dynamics is presented in [25], which is able to solve time-varying nonlinear optimization (termed future dynamic nonlinear optimization [FDNO] problem) accurately without perturbed by noises. However, in spite of the fact that noises and perturbations are widely existing in the online solution process, existing methods for solving the FDNO problem in the presence of noises are considerably rare. Therefore, it is of crucial importance to find a new computational method to handle noises and perturbations with high accuracy achieved for the FDNO problem.

Considering that the continuous-time model cannot be applied to digital hardware directly, the computational method depicted in discrete form is desirable. To this end, based on neural dynamics, a discrete-time noise-tolerant neural dynamics (NTND) model is constructed in this chapter to solve the FDNO problem in the presence of noises and perturbations. Given that the Hessian matrix inversion is involved in the NTND model, the Broyden-Fletcher-Goldfarb-Shanno (BFGS) [28–30] is leveraged to approximate the inverse of the Hessian matrix, which is especially helpful for the situation that direct computing of the Hessian matrix inverse is expensive or difficult to conduct. The content of this chapter is organized as follows. In Sect. 3.2, the FDNO problem is formulated and the NTND and NTND-BFGS models are given to handle such a future problem. Moreover, for comparison, the existing solutions are presented as well. Then, Sect. 3.3 provides theoretical analyses to illustrate the global convergence of the presented NTND and NTND-BFGS models with or without noises. Moreover, numerical experiments and applications to the robot manipulator are presented in Sect. 3.4 to validate the superiority of the constructed NTND and NTND-BFGS models, as compared with other existing models. Finally, conclusions are drawn in Sect. 3.6. In the end of this introductory section, the main contributions of this chapter are summarized as follows:

- (1) This is the first work for solving nonlinear optimizations with dynamic parameters and noise suppression, of which an intrinsic requirement is that the solution should be calculated before its corresponding mathematical formulation appears. In this sense, this is quite different from the conventionally investigated static optimization and thus termed the future dynamic nonlinear optimization (FDNO) problem.
- (2) Two neural dynamics model, termed NTND and NTND-BFGS, are designed to solve the FDNO problems in the presence of noises based on the neural dynamics approach, of which the latter one eliminates the intensively computational burden for matrix inversion.
- (3) Control techniques are leveraged to conduct the theoretical analyses, which reveal that the residual errors of the two given neural dynamics models are able to converge to a tiny value near zero globally with or without noises.

3.2 Problem Formulation and Solutions

This section presents the framework and formulations of the FDNO problem with two newly presented neural dynamics models. For comparison, existing solutions are provided as well.

3.2.1 Problem Formulation

It is required in the digital hardware implementation that a problem should be depicted in discrete form. Therefore, it is desirable to formulate a problem in discrete manner. Let t_s and t_f denote the start and the final time instant of the solving process, respectively. An FDNO problem, for which the calculation should be conducted during the time interval $[t_k, t_{k+1}] \in [t_s, t_f]$, is expressed as

$$\min_{\mathbf{y}(t_{k+1}) \in \mathbb{R}^m} \Phi(\mathbf{y}(t_{k+1}), t_{k+1}) \in \mathbb{R}, \quad (3.1)$$

where $t = k\delta$ with updating index $k = 0, 1, 2, \dots$, which is abbreviated as t_k ; $\delta > 0$ represents the time sampling gap; $\Phi(\mathbf{y}(t_{k+1}), t_{k+1})$ is discretized from the smoothly time-varying signal $\Phi(\mathbf{y}(t), t)$, for which the following assumptions are made: $\Phi(\cdot, \cdot)$ is a time-varying nonlinear function and twice differentiable and lower bounded.

This work is dedicated to find the future solution $\mathbf{y}(t_{k+1}) \in \mathbb{R}^m$ during the computational interval $[t_k, t_{k+1}]$ that makes function (3.1) achieve its minimum value at time instant t_{k+1} . Note that, during the present computational interval $[t_k, t_{k+1}]$, $\Phi(\mathbf{y}(t_{k+1}), t_{k+1})$ and its derivatives are not available. In this sense, only the present and/or previous data (e.g., $\mathbf{y}(t_k)$) rather than the unknown data (e.g., $\mathbf{y}(t_{k+1})$) can be leveraged to compute $\mathbf{y}(t_{k+1})$.

3.2.2 Continuous-Time NTND Model

The continuous-time FDNO problem is defined as

$$\min_{\mathbf{y}(t) \in \mathbb{R}^m} \Phi(\mathbf{y}(t), t) \in \mathbb{R}, t \in [t_s, t_e], \quad (3.2)$$

of which the gradient is

$$\mathbf{q}(\mathbf{y}(t), t) = \frac{\partial \Phi(\mathbf{y}(t), t)}{\partial \mathbf{y}(t)} \in \mathbb{R}^m. \quad (3.3)$$

In addition, the time derivative of the above equation can be written as

$$\begin{aligned}\frac{d\mathbf{q}(\mathbf{y}(t), t)}{dt} &= \frac{\partial \mathbf{q}(\mathbf{y}(t), t)}{\partial t} + \frac{\partial \mathbf{q}(\mathbf{y}(t), t)}{\partial \mathbf{y}(t)} \frac{d\mathbf{y}(t)}{dt} \\ &= \dot{\mathbf{q}}_t(\mathbf{y}(t), t) + H(\mathbf{y}(t), t) \frac{d\mathbf{y}(t)}{dt},\end{aligned}\quad (3.4)$$

where $H(\mathbf{y}(t), t) \in \mathbb{R}^{m \times m}$ represents the Hessian matrix. In detail, $\dot{\mathbf{q}}_t(\mathbf{y}(t), t)$ is the derivative of $\mathbf{q}(\mathbf{y}(t), t)$ with respect to time t and can be defined as

$$\dot{\mathbf{q}}_t(\mathbf{y}(t), t) = \frac{\partial \mathbf{q}(\mathbf{y}(t), t)}{\partial t} = \frac{\partial^2 \Phi(\mathbf{y}(t), t)}{\partial \mathbf{y}(t) \partial t} \in \mathbb{R}^m.$$

It has been proven in [25] that the FDNO problem achieves its minimal if $\mathbf{q}(\mathbf{y}(t), t) = 0$ with a position definite $H(\mathbf{y}(t), t)$.

For performance evaluations in this chapter, how well each model solves the FDNO problem is observed through the following error function:

$$\boldsymbol{\xi}(t) = [\xi_1(t), \xi_2(t), \dots, \xi_m(t)]^T = \mathbf{q}(\mathbf{y}(t), t) \in \mathbb{R}^m, \quad (3.5)$$

where $\xi_h(t)$ is the h th element of $\boldsymbol{\xi}(t)$, $\forall h \in \{1, 2, \dots, m\}$.

Based on the design formula constructed in [31, 32]: $\dot{\boldsymbol{\xi}}(t) = -\gamma \boldsymbol{\xi}(t) - \lambda \int_0^t \boldsymbol{\xi}(\alpha) d\alpha$, a continuous-time NTND (CT-NTND) model perturbed by noises is designed as

$$\begin{aligned}H(\mathbf{y}(t), t) \dot{\mathbf{y}}(t) &= -\gamma \mathbf{q}(t) - \dot{\mathbf{q}}_t(\mathbf{y}(t), t) - \\ &\quad \lambda \int_0^t \mathbf{q}(\mathbf{y}(\alpha), \alpha) d\alpha + \varepsilon(t),\end{aligned}\quad (3.6)$$

where $\lambda > 0$ and $\gamma > 0$ with an invertible Hessian matrix; $\varepsilon(t)$ represents additive noise in the system, which can be classified as constant noises, linear noises, and random noises. Then, the CT-NTND model can be formulated as

$$\begin{aligned}\dot{\mathbf{y}}(t) &= -H^{-1}(\mathbf{y}(t), t) (\gamma \mathbf{q}(\mathbf{y}(t), t) + \dot{\mathbf{q}}_t(\mathbf{y}(t), t) + \\ &\quad \lambda \int_0^t \mathbf{q}(\mathbf{y}(\alpha), \alpha) d\alpha + \varepsilon(t)) \\ &= -H^{-1}(\mathbf{y}(t), t) \left(\gamma \left(\frac{\partial \Phi(\mathbf{y}(t), t)}{\partial \mathbf{y}(t)} \right) + \frac{\partial^2 \Phi(\mathbf{y}(t), t)}{\partial \mathbf{y}(t) \partial t} + \right. \\ &\quad \left. \lambda \int_0^t \frac{\partial \Phi(\mathbf{y}(\alpha), \alpha)}{\partial \mathbf{y}(\alpha)} d\alpha + \varepsilon(t) \right).\end{aligned}\quad (3.7)$$

3.2.3 Existing Discrete-Time Solutions

The Euler forward difference formula is always exploited to discretize continuous-time models:

$$\dot{\mathbf{y}}(t_k) \approx \frac{\mathbf{y}(t_{k+1}) - \mathbf{y}(t_k)}{\delta}. \quad (3.8)$$

Thus, the discrete-time neural dynamics (DTND) model derived from continuous-time neural dynamics model in [31, 32] is formulated as

$$\begin{aligned} \mathbf{y}(t_{k+1}) = & \mathbf{y}(t_k) - H^{-1}(\mathbf{y}(t_k), t_k)((c+1)\mathbf{q}(\mathbf{y}(t_k), t_k) - \\ & \mathbf{q}(\mathbf{y}(t_k), t_{k-1})), \end{aligned} \quad (3.9)$$

where step size $c = \delta\gamma > 0$. It is obvious that without $\mathbf{q}(\mathbf{y}(t_0), t_{-1})$, we cannot get $\dot{\mathbf{q}}_t(\mathbf{y}(t_0), t_0)$ through (3.8). Therefore, by setting $\dot{\mathbf{q}}_t(\mathbf{y}(t_0), t_0) = 0$, the initial update is formulated as $\mathbf{y}(t_1) = \mathbf{y}(t_0) - cH^{-1}(\mathbf{y}(t_0), t_0)\mathbf{q}(\mathbf{y}(t_0), t_0)$.

Subsequently, different DTND models can be obtained by means of numerical formulas presented in [33], of which the three-step DTND model is defined as

$$\begin{aligned} \mathbf{y}(t_{k+1}) = & \frac{3}{2}\mathbf{y}(t_k) - \mathbf{y}(t_{k-1}) + \frac{1}{2}\mathbf{y}(t_{k-2}) - \\ & H^{-1}(\mathbf{y}(t_k), t_k)((c+1)\mathbf{q}(\mathbf{y}(t_k), t_k) - \\ & \mathbf{q}(\mathbf{y}(t_k), t_{k-1})), \end{aligned} \quad (3.10)$$

and the four-step DTND model is obtained

$$\begin{aligned} \mathbf{y}(t_{k+1}) = & \frac{-1}{8}\mathbf{y}(t_k) + \frac{3}{4}\mathbf{y}(t_{k-1}) + \frac{5}{8}\mathbf{y}(t_{k-2}) - \frac{1}{4}\mathbf{y}(t_{k-3}) \\ & - \frac{9}{4}H^{-1}(\mathbf{y}(t_k), t_k)((c+1)\mathbf{q}(\mathbf{y}(t_k), t_k) - \\ & \mathbf{q}(\mathbf{y}(t_k), t_{k-1})). \end{aligned} \quad (3.11)$$

In addition, the five-step DTND model [34] is also presented here:

$$\begin{aligned} \mathbf{y}(t_{k+1}) = & \frac{6}{13}\mathbf{y}(t_k) + \frac{2}{13}\mathbf{y}(t_{k-1}) + \frac{4}{13}\mathbf{y}(t_{k-2}) + \\ & \frac{3}{13}\mathbf{y}(t_{k-3}) - \frac{2}{13}\mathbf{y}(t_{k-4}) - \\ & \frac{24}{13}H^{-1}(\mathbf{y}(t_k), t_k)((c+1)\mathbf{q}(\mathbf{y}(t_k), t_k) - \\ & \mathbf{q}(\mathbf{y}(t_k), t_{k-1})). \end{aligned} \quad (3.12)$$

Besides, the NRI model [35] is provided as follows:

$$\mathbf{y}(t_{k+1}) = \mathbf{y}(t_k) - H^{-1}(\mathbf{y}(t_k), t_k) \mathbf{q}(\mathbf{y}(t_k), t_k). \quad (3.13)$$

3.2.4 NTND and NTND-BFGS Models

Noise interference is ever present during the solving process, e.g., the observational error, the truncation error, the quantization error, and the sampling error. Therefore, a high-speed algorithm with noise-tolerant competence for solving the FDNO problem is in demand. In this section, discrete-time NTND and NTND-BFGS models are derived to tolerate noises during the solution.

In order to simplify the structure of the discrete-time NTND model, the numerical differentiation formula with the least items is chosen. Thus, substituting Euler forward difference (3.8) to CT-NTND model (3.7), we can obtain the discrete-time NTND model:

$$\begin{aligned} \mathbf{y}(t_{k+1}) = & \mathbf{y}(t_k) - H^{-1}(\mathbf{y}(t_k), t_k) \left((c_1 + 1) \mathbf{q}(\mathbf{y}(t_k), t_k) - \right. \\ & \left. \mathbf{q}(\mathbf{y}(t_k), t_{k-1}) + c_2 \sum_{j=0}^k \mathbf{q}(\mathbf{y}(t_j), t_j) \right), \end{aligned} \quad (3.14)$$

where step size $c_1 = \delta\gamma > 0$ and $c_2 = \delta\lambda > 0$; the noise term $\varepsilon(t_k)$ is eliminated to depict the NTND model structure only. The sum term of $\mathbf{q}(\mathbf{y}(t_j), t_j)$, which is discretized from the integral term of (3.7), plays a significant role in offsetting the impact brought by abrupt disturbance for (3.14). The circuit diagram showing general components of NTND model (3.14) is illustrated in Fig. 3.1.

To solve $\mathbf{y}(t_{k+1})$ through NTND model (3.14), the calculation of Hessian matrix inversion is unavoidable, which could be quite costly if the Hessian matrix is complicated. Besides, it is not likely to make the calculation offline because $H^{-1}(\mathbf{y}(t_k), t_k)$ is required to be computed online. In order to overcome this drawback, the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm [36] is utilized in this section.

The highlight of BFGS algorithm [36] is its capability to explicitly escape the usage of the inverted Hessian matrix, reducing the computation complexity. In BFGS, exact Hessian matrix is replaced by an approximation consisting of least-change updates generated from the gradient at every iteration. It is known in Sect. 3.2 that $\Phi(\cdot, \cdot)$ is a convex function whose Hessian matrix is positive definite. Thus, we can conclude that the Hessian approximation matrix obtained by BFGS algorithm [36] converges to the Hessian matrix inversion.

The following NTND-BFGS model is given for FDNO (3.1) in case of computing the Hessian matrix inversion being expensive:

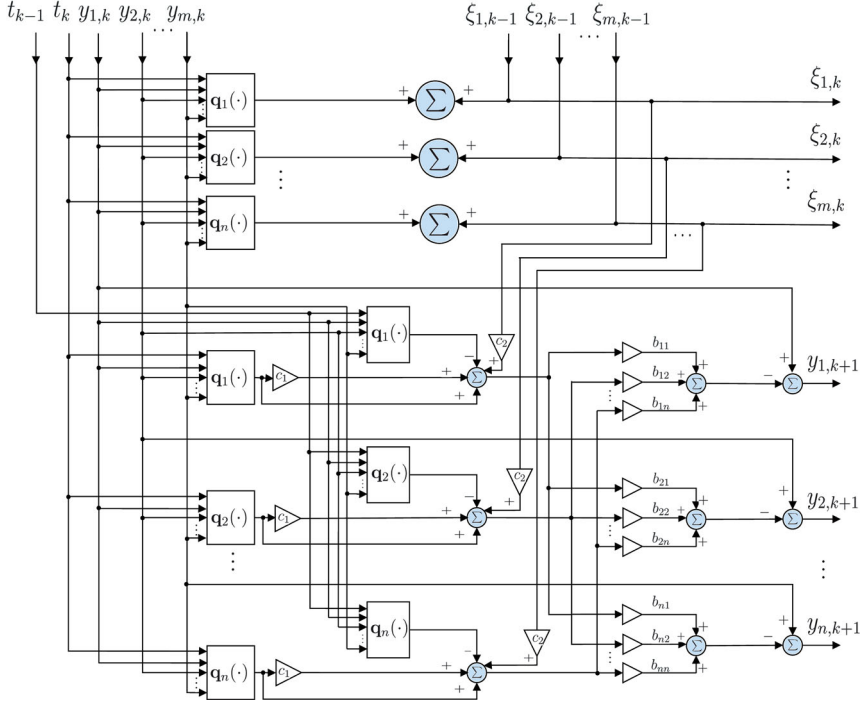


Fig. 3.1 Schematic circuit diagram realizing NTND model (3.14) for solving FDNO problem (3.1), where (3.14) can be rewritten as the following two equations: $\mathbf{y}(t_{k+1}) = \mathbf{y}(t_k) - H^{-1}(\mathbf{y}(t_k), t_k)((c_1 + 1)\mathbf{q}(\mathbf{y}(t_k), t_k) - \mathbf{q}(\mathbf{y}(t_k), t_{k-1}) + c_2\boldsymbol{\xi}(t_k))$ and $\boldsymbol{\xi}(t_k) = \boldsymbol{\xi}(t_{k-1}) + \mathbf{q}(\mathbf{y}(t_k), t_k)$, with $B = H^{-1}(\mathbf{y}(t_k), t_k)$ and its $\sigma\zeta$ th entry denoted by $b_{\sigma\zeta}$

$$\mathbf{y}(t_{k+1}) = \mathbf{y}(t_k) - \tilde{D}_k(\mathbf{y}(t_k), t_k)((c_1 + 1)\mathbf{q}(\mathbf{y}(t_k), t_k) - \mathbf{q}(\mathbf{y}(t_k), t_{k-1}) + c_2 \sum_{j=0}^k \mathbf{q}(\mathbf{y}(t_j), t_j)), \quad (3.15)$$

where $\tilde{D}_k(\mathbf{y}(t_k), t_k)$ is the approximation of $H^{-1}(\mathbf{y}(t_k), t_k)$ supported by BFGS iterative formula:

$$\tilde{D}_{k+1} = \tilde{D}_k + \Delta \tilde{D}_k, \quad (3.16)$$

where

$$\Delta \tilde{D}_k = \frac{(s_k^T z_k + z_k^T \tilde{D}_k z_k)(s_k s_k^T)}{(s_k^T z_k)^2} - \frac{(\tilde{D}_k z_k s_k^T + s_k z_k^T \tilde{D}_k)}{(s_k^T z_k)}; \quad (3.17)$$

$\mathbf{y}(t_{k+1}) = \mathbf{y}(t_k) + s_k$; $\tilde{D}_k(\mathbf{y}(t_{k+1}) - \mathbf{y}(t_k)) = z_k$. Computing (3.17) has no temporary matrices. Besides, scalar $s_k^T z_k$, $z_k^T \tilde{D}_k z_k$ and symmetric $\tilde{D}_k$ accelerate the computation. The initial iterative value $\tilde{D}_0$ should be positive definite to achieve rapid convergence, wherein $\tilde{D}_0 = I$ is a typical choice.

3.3 Theoretical Analyses and Results

NTND model (3.14) polluted by noises is written as

$$\begin{aligned} \mathbf{y}(t_{k+1}) = & \mathbf{y}(t_k) - H^{-1}(\mathbf{y}(t_k), t_k)((c_1 + 1)\mathbf{q}(\mathbf{y}(t_k), t_k) - \\ & \mathbf{q}(\mathbf{y}(t_k), t_{k-1}) + c_2 \sum_{j=0}^k \mathbf{q}(\mathbf{y}(t_j), t_j) + \varepsilon(t_k)). \end{aligned} \quad (3.18)$$

Theorem 3.1 *There is an equivalence between NTND model (3.14) and the following equation:*

$$\xi(t_{k+1}) + (c_1 - 1)\xi(t_k) + c_2 \sum_{j=0}^k \xi(t_j) + \mathbf{O}(\delta^2) = 0, \quad (3.19)$$

where $\mathbf{O}(\delta^2)$ denotes the vector of truncation errors with each entry being $O(\delta^2)$.

Proof Equation (3.14) can be rewritten as

$$\begin{aligned} H(\mathbf{y}(t_k), t_k)(\mathbf{y}(t_{k+1}) - \mathbf{y}(t_k)) = & -(c_1 + 1)\mathbf{q}(\mathbf{y}(t_k), t_k) + \\ & \mathbf{q}(\mathbf{y}(t_k), t_{k-1}) - c_2 \sum_{j=0}^k \mathbf{q}(\mathbf{y}(t_j), t_j). \end{aligned} \quad (3.20)$$

Then, we can obtain

$$\begin{aligned} & -c_1 \mathbf{q}(\mathbf{y}(t_k), t_k) - c_2 \sum_{j=0}^k \mathbf{q}(\mathbf{y}(t_j), t_j) \\ & = H(\mathbf{y}(t_k), t_k)(\mathbf{y}(t_{k+1}) - \mathbf{y}(t_k)) + (\mathbf{q}(\mathbf{y}(t_k), t_k) - \\ & \quad \mathbf{q}(\mathbf{y}(t_k), t_{k-1})). \end{aligned} \quad (3.21)$$

The Euler forward difference formula is utilized for

$$\mathbf{y}(t_{k+1}) - \mathbf{y}(t_k) = \delta \dot{\mathbf{y}}(t_k) + \mathbf{O}(\delta^2),$$

and

$$\mathbf{q}(\mathbf{y}(t_k), t_k) - \mathbf{q}(\mathbf{y}(t_{k-1}), t_{k-1}) = \delta \dot{\mathbf{q}}(\mathbf{y}(t_k), t_k) + \mathbf{O}(\delta^2).$$

Substituting the above two formulas into (3.21) can directly generate

$$\begin{aligned} & H(\mathbf{y}(t_k), t_k)(\delta \dot{\mathbf{y}}(t_k) + \mathbf{O}(\delta^2)) + \delta \dot{\mathbf{q}}(\mathbf{y}(t_k), t_k) + \mathbf{O}(\delta^2) = \\ & -c_1 \mathbf{q}(\mathbf{y}(t_k), t_k) - c_2 \sum_{j=0}^k \mathbf{q}(\mathbf{y}(t_j), t_j). \end{aligned} \quad (3.22)$$

The discrete version of equation (3.4) is

$$\dot{\boldsymbol{\xi}}(t_k) = \dot{\mathbf{q}}_r(\mathbf{y}(t_k), t_k) + H(\mathbf{y}(t_k), t_k) \dot{\mathbf{y}}(t_k). \quad (3.23)$$

Then, substituting (3.23) into (3.22), a simple form of (3.22) with respect to the error function is shown as

$$\delta \dot{\boldsymbol{\xi}}(t_k) + \mathbf{O}(\delta^2) = -c_1 \boldsymbol{\xi}(t_k) - c_2 \sum_{j=0}^k \boldsymbol{\xi}(t_j). \quad (3.24)$$

In addition, operating Euler forward difference formula (3.8) on $\boldsymbol{\xi}(\mathbf{y}(t_k), t_k)$ leads to

$$\delta \frac{\boldsymbol{\xi}(t_{k+1}) - \boldsymbol{\xi}(t_k)}{\delta} + \mathbf{O}(\delta^2) = -c_1 \boldsymbol{\xi}(t_k) - c_2 \sum_{j=0}^k \boldsymbol{\xi}(t_j), \quad (3.25)$$

which is simplified as

$$\boldsymbol{\xi}(t_{k+1}) + (c_1 - 1)\boldsymbol{\xi}(t_k) + c_2 \sum_{j=0}^k \boldsymbol{\xi}(t_j) + \mathbf{O}(\delta^2) = 0. \quad (3.26)$$

The proof is complete. $\square$

Regarding the noise-suppressing property of NTND model (3.14), we offer the following theoretical analyses.

Theorem 3.2 Consider FDNO (3.1). The residual error $\lim_{k \rightarrow \infty} \|\boldsymbol{\xi}(t_k)\|_2$ of NTND model (3.14) for solving future dynamic nonlinear optimization problem is $O(\delta^2)$, wherein $\|\cdot\|_2$ denotes 2-norm.

Proof Using $\boldsymbol{\xi}^i(t_k)$ to denote the i th subsystem of $\boldsymbol{\xi}(t_k)$ generates

$$\boldsymbol{\xi}^i(t_k) + (c_1 - 1)\boldsymbol{\xi}^i(t_{k-1}) + c_2 \sum_{j=0}^{k-1} \boldsymbol{\xi}^i(t_j) + O(\delta^2) = 0. \quad (3.27)$$

Subtracting (3.27) from the i th subsystem of (3.19), we can get

$$\xi^i(t_{k+1}) = (2 - c_1 - c_2)\xi^i(t_k) + (c_1 - 1)\xi^i(t_{k-1}) + O(\delta^2). \quad (3.28)$$

Letting $\Upsilon^i(t_k) = [\xi^i(t_{k+1}), \xi^i(t_k)]^T$, we can transform equation (3.28) into the pattern of the state-space matrix:

$$\Upsilon^i(t_{k+1}) = U\Upsilon^i(t_k) + \mathbf{O}(\delta^2), \quad (3.29)$$

where

$$U = \begin{bmatrix} (2 - c_1 - c_2) & (c_1 - 1) \\ 1 & 0 \end{bmatrix}. \quad (3.30)$$

Taking (3.29) into Minkowski's inequality [37] for the 2-norm, we have

$$\begin{aligned} \|\Upsilon^i(t_{k+1})\|_2 &\leq \|U\Upsilon^i(t_k)\|_2 + \|\mathbf{O}(\delta^2)\|_2 \\ &= \|U\Upsilon^i(t_k)\|_2 + O(\delta^2) \\ &\leq \|U\|_2 \|\Upsilon^i(t_k)\|_2 + O(\delta^2) \\ &\leq \|U\|_2 \|\Upsilon^i(t_{k-1})\|_2 + \|U\|_2 O(\delta^2) + O(\delta^2) \\ &= \|U\|_2 \|U\Upsilon^i(t_{k-1})\|_2 + O(\delta^2) \\ &\leq \|U\|_2^2 \|\Upsilon^i(t_{k-1})\|_2 + O(\delta^2) \\ &\quad \vdots \\ &\leq \|U\|_2^k \|\Upsilon^i(t_1)\|_2 + O(\delta^2). \end{aligned} \quad (3.31)$$

Matrix U (3.30) has two different eigenvalues which are $\mu_1 = 1 - (c_1 + c_2 + \sqrt{(c_1 + c_2)^2 - 4c_2})/2$ and $\mu_2 = 1 - (c_1 + c_2 - \sqrt{(c_1 + c_2)^2 - 4c_2})/2$. The region shown in Fig. 3.2 describes the value of c_1, c_2 that make the real part of μ_1, μ_2 ranging from -1 to 1 . Thereby, we can obtain $\lim_{k \rightarrow \infty} \|U\|_2^k = 0$ which further assist that

$$\lim_{k \rightarrow \infty} \|\Upsilon^i(t_{k+1})\|_2 \leq \lim_{k \rightarrow \infty} \|U\|_2^k \|\Upsilon^i(t_1)\|_2 + O(\delta^2) = O(\delta^2). \quad (3.32)$$

As long as c_1, c_2 which we choose for (3.14) belonging to that parameter region of convergence (PROC) depicted in Fig. 3.2, we have $\lim_{k \rightarrow \infty} \|\xi(t_k)\|_2$ of NTND model (3.14) convergent to $O(\delta^2)$. The proof is complete. $\square$

Theorem 3.3 Consider FDNO (3.1). Polluted by arbitrary linear noise $\varepsilon(t_k) = \eta\delta k + \Omega$, the residual error $\lim_{k \rightarrow \infty} \|\xi(t_k)\|_2$ of NTND model (3.14) for solving the future dynamic nonlinear optimization problem is $\|\eta\delta/c_2\|_2 + O(\delta^2)$; for constant

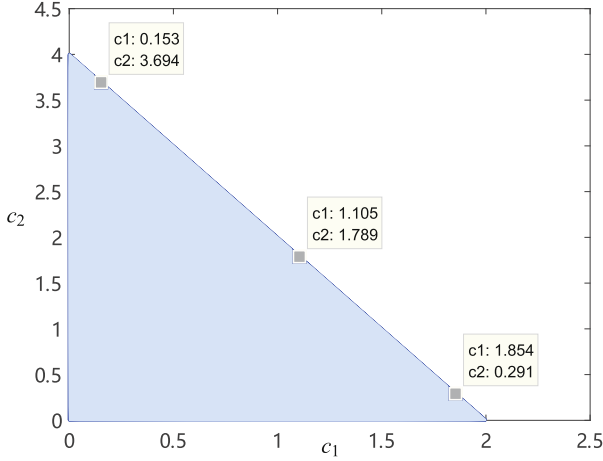


Fig. 3.2 Parameter region of convergence (PROC) on NTND model (3.14), where the PROC upper border lies in function $c_2 = -2c_1 + 4$ which is approximated by the least square method; c_1 and c_2 in the picture correspond to step size c_1, c_2 , respectively

noise $\varepsilon(t_k) = \Omega$ degraded from linear noises with $\eta = 0$, the residual error of NTND model (3.14) is $O(\delta^2)$, which has nothing to do with the value of constant noise Ω .

Proof The factors that determine the residual error of noise-polluted NTND model (3.18) can be classified as $O(\delta^2)$ and injecting constant noise $\varepsilon(t_k)$. The linearity property of (3.19) allows us to independently investigate the two factors.

Firstly, we simply change the form of (3.19) as

$$\xi(t_{k+1}) = ((1 - c_1)\xi(t_k) - c_2 \sum_{j=0}^k \xi(t_j)) + O(\delta^2). \quad (3.33)$$

It has been proven in Theorem 3.2 that the residual error of (3.33) is $O(\delta^2)$.

Next, consider how constant noises influence the convergent performance of

$$\xi(t_{k+1}) + (c_1 - 1)\xi(t_k) + c_2 \sum_{j=0}^k \xi(t_j) + \varepsilon(t_k) = 0. \quad (3.34)$$

In a more general sense, constant noise $\varepsilon(t_k) = \Omega$ is a subcase of linear time-variant noise $\varepsilon(t_k) = \eta\delta k + \Omega, k = 0, 1, 2, \dots$. Thus, the subsystem of (3.34) can be rewritten as

$$\xi^i(t_{k+1}) = ((1 - c_1)\xi(t_k) - c_2 \sum_{j=0}^k \xi(t_j)) + \eta^i k \delta + \Omega^i. \quad (3.35)$$

The Z-transformation of (3.35) is

$$\xi^i(z) = \frac{-\Omega^i(z-1)^2 - \eta^i \delta z + z(z-1)^2 \xi^i(0)}{z(z-1)^2 + (c_1 - 1)(z-1)^2 + c_2 z(z-1)}, \quad (3.36)$$

where $z_1, z_2 = 1 - (c_1 + c_2 \pm \sqrt{(c_1 + c_2)^2 - 4c_2})/2$ are poles of $\xi^i(z)$, with $\xi^i(0)$ as its initiation. It is the same as the range of c_1, c_2 in Fig. 3.2. Thus, utilizing the Z-transformation final theorem of (3.36), the limit formula of $\xi^i(t_k)$ is

$$\begin{aligned} & \lim_{k \rightarrow \infty} \xi^i(t_k) \\ &= \lim_{z \rightarrow 1} (z-1) \xi^i(z) \\ &= \lim_{z \rightarrow 1} \frac{-\Omega^i(z-1)^2 - \eta^i \delta z + z(z-1)^2 \xi^i(0)}{z(z-1) + (c_1 - 1)(z-1) + c_2 z} \\ &= \frac{-\eta^i \delta}{c_2}. \end{aligned} \quad (3.37)$$

Therefore, as a result of the effect of linear noise $\varepsilon(t_k) = \eta \delta k + \Omega$, the residual error of (3.34) is

$$\lim_{k \rightarrow \infty} \|\xi(t_k)\|_2 = \left\| \frac{\eta \delta}{c_2} \right\|_2.$$

Setting $\eta = 0$, we can instantly know that when it comes to constant noise $\varepsilon(t_k) = \Omega$, the residual error of (3.34) is

$$\lim_{k \rightarrow \infty} \|\xi(t_k)\|_2 = O(\delta).$$

Summing up, taking constant noises and the residual error of (3.33) into account, it is for certain that the residual error $\lim_{k \rightarrow \infty} \|\xi(t_k)\|_2$ of NTND model (3.18) is $O(\delta)$, which has nothing to do with the value of constant noises. The proof is complete. $\square$

For further investigation, the ensuring theorem reveals how NTND model (3.14) handles bounded random noises.

Theorem 3.4 Consider FDNO (3.1). For bounded random noise $\varepsilon(t) = \rho$, NTND model (3.14) keeps a bounded residual error $\lim_{k \rightarrow \infty} \|\xi(t_k)\|_2$, which is $2m \sup_{1 \leq i \leq k, 1 \leq i \leq m} |\rho_i^i| / (1 - \|U\|_2) + O(\delta^2)$.

Proof Similar with the superposition principle exploited in Theorem 3.3, primarily, the difference equation is generated:

$$\begin{aligned} \xi^i(t_{k+1}) = & (2 - c_1 - c_2)\xi^i(t_k) + (c_1 - 1)\xi^i(t_{k-1}) + \\ & \rho^i(t_k) - \rho^i(t_{k-1}). \end{aligned} \quad (3.38)$$

Let $\varpi^i(t_k) = [\rho^i(t_k) - \rho^i(t_{k-1}), 0]^T$, the aforesaid (3.38) in terms of $\Upsilon^i(t_k)$ is

$$\Upsilon^i(t_{k+1}) = U\Upsilon^i(t_k) + \varpi^i(t_k), \quad (3.39)$$

where

$$U = \begin{bmatrix} (2 - c_1 - c_2) & (c_1 - 1) \\ 1 & 0 \end{bmatrix}; \quad (3.40)$$

then it is of foundation to obtain

$$\begin{aligned} & \|\Upsilon^i(t_{k+1})\|_2 \\ & \leq \|U\Upsilon^i(t_k)\|_2 + \|\varpi^i(t_k)\|_2 \\ & \leq \|U\|_2 \|\Upsilon^i(t_k)\|_2 + \|\varpi^i(t_k)\|_2 \\ & \leq \|U\|_2 \|U\Upsilon^i(t_{k-1})\|_2 + \|U\|_2 \|\varpi^i(t_{k-1})\|_2 + \|\varpi^i(t_k)\|_2 \\ & \leq \|U\|_2^2 \|\Upsilon^i(t_{k-1})\|_2 + \|U\|_2 \|\varpi^i(t_{k-1})\|_2 + \|\varpi^i(t_k)\|_2 \\ & \quad \vdots \\ & \leq \|U\|_2^k \|\Upsilon^i(t_1)\|_2 + \|U\|_2^{k-1} \|\varpi^i(t_1)\|_2 + \dots + \|\varpi^i(t_k)\|_2 \\ & < \|U\|_2^k \|\Upsilon^i(t_1)\|_2 + \max_{1 \leq t \leq k} \|\varpi_t^i\|_2 / (1 - \|U\|_2) \\ & < \|U\|_2^k \|\Upsilon^i(t_1)\|_2 + 2 \max_{1 \leq t \leq k} |\rho_t^i| / (1 - \|U\|_2). \end{aligned} \quad (3.41)$$

Moreover, with $\lim_{k \rightarrow \infty} \|U\|_2^k = 0$, we can get

$$\lim_{k \rightarrow \infty} \|\Upsilon_{k+1}^i\|_2 < 2 \max_{1 \leq t \leq k} |\rho_t^i| / (1 - \|U\|_2).$$

Finally, we come to the conclusion that

$$\lim_{k \rightarrow \infty} \|\xi(t_k)\|_2 < 2m \sup_{1 \leq t \leq k, 1 \leq i \leq m} |\rho_t^i| / (1 - \|U\|_2) + O(\delta^2).$$

The proof is complete. $\square$

The BFGS algorithm has been found dramatically helpful for easing the expensive computational burden for NTND model (3.14), while the Hessian matrix inversion is difficult to obtain. Several theorems regarding NTND-BFGS model (3.15) are provided as follows. In addition, noises denote by $\varepsilon(t)$ polluted NTND-BFGS model (3.15)

$$\begin{aligned} \mathbf{y}(t_{k+1}) &= \mathbf{y}(t_k) - \tilde{D}_k(\mathbf{y}(t_k), t_k)((c_1 + 1)\mathbf{q}(\mathbf{y}(t_k), t_k) \\ &\quad - \mathbf{q}(\mathbf{y}(t_k), t_{k-1}) + c_2 \sum_{j=0}^k \mathbf{q}(\mathbf{y}(t_j), t_j) + \varepsilon(t_k)). \end{aligned} \quad (3.42)$$

Theorem 3.5 Consider FDNO (3.1). Given that $\|\tilde{D}_k - H_k^{-1}\|_2 = O(\delta^2)$, the theoretical solution generated by NTND-BFGS model (3.15) converges to that solution generated by NTND model (3.14) with the residual error being $O(\delta^2)$.

Proof To begin this proof, $\tilde{\mathbf{y}}(t_k)$ and $\mathbf{y}(t_k)$ are used to denote the solutions of NTND-BFGS model (3.15) and NTND model (3.14), respectively. We further substitute $\|\tilde{D}_k - H_k^{-1}\|_2 = O(\delta^2)$ into (3.15) and then obtain

$$\begin{aligned} \tilde{\mathbf{y}}(t_{k+1}) &= \mathbf{y}(t_k) - (H_k^{-1} + O(\delta^2))((c_1 + 1)\mathbf{q}(\mathbf{y}(t_k), t_k) - \\ &\quad \mathbf{q}(\mathbf{y}(t_k), t_{k-1}) + c_2 \sum_{j=0}^k \mathbf{q}(\mathbf{y}(t_j), t_j)) \\ &= \mathbf{y}(t_k) - H_k^{-1}((c_1 + 1)\mathbf{q}(\mathbf{y}(t_k), t_k) - \\ &\quad \mathbf{q}(\mathbf{y}(t_k), t_{k-1}) + c_2 \sum_{j=0}^k \mathbf{q}(\mathbf{y}(t_j), t_j)) - \\ &\quad O(\delta^2)((c_1 + 1)\mathbf{q}(\mathbf{y}(t_k), t_k) - \mathbf{q}(\mathbf{y}(t_k), t_{k-1}) + c_2 \sum_{j=0}^k \mathbf{q}(\mathbf{y}(t_j), t_j)) \\ &= \mathbf{y}(t_k) - (H_k^{-1}((c_1 + 1)\mathbf{q}(\mathbf{y}(t_k), t_k) - \\ &\quad \mathbf{q}(\mathbf{y}(t_k), t_{k-1}) + c_2 \sum_{j=0}^k \mathbf{q}(\mathbf{y}(t_j), t_j)) + O(\delta^2) \\ &= \mathbf{y}(t_{k+1}) + O(\delta^2). \end{aligned} \quad (3.43)$$

It is evident that

$$\tilde{\mathbf{y}}(t_{k+1}) = \mathbf{y}(t_{k+1}) + O(\delta^2).$$

Therefore, we can get the residual error relationship between NTND-BFGS model (3.15) and NTND model (3.14):

$$\|\tilde{\mathbf{y}}(t_k) - \mathbf{y}(t_k)\|_2 = O(\delta^2).$$

The proof is complete. $\square$

Theorem 3.5 verifies the feasibility of using NTND-BFGS model (3.15) to replace the inverse calculation from the perspective of theoretical derivation. In addition, the following theorem is provided to reveal the effectiveness of the NTND-BFGS model (3.15).

Theorem 3.6 *Consider FDNO (3.1).*

- (1) *When it comes to NTND-BFGS model (3.15) for solving the future dynamic nonlinear optimization problem, the residual error $\lim_{k \rightarrow \infty} \|\xi(t_k)\|_2$ is $O(\delta^2)$.*
- (2) *With respect to noise-polluted NTND-BFGS model (3.42) for solving the future dynamic nonlinear optimization problem, the residual error $\lim_{k \rightarrow \infty} \|\xi(t_k)\|_2$ is $O(\delta^2)$ which has nothing to do with the value of constant noise $\varepsilon(t)$.*
- (3) *For a bounded random noise $\varepsilon(t) = \rho$, the residual error of NTND-BFGS model (3.42) is bounded, with its upper bound $\lim_{k \rightarrow \infty} \|\xi(t_k)\|_2$ being $2m \sup_{1 \leq t \leq k, 1 \leq i \leq m} |\rho_t^i| / (1 - \|U\|_2) + O(\delta^2)$.*

Proof It can be generalized from proofs in Theorem 3.2 though Theorem 3.4 and thus omitted. The proof is complete. $\square$

Via the theorem constructed in [25], the residual error of traditional methods which are intrinsically constructed to solve static optimization problems for future dynamic nonlinear optimization is $O(\delta)$, within computing interval $[0, \delta)$.

3.4 Numerical Experiments

In this section, the effectiveness of the designed NTND model (3.14) and NTND-BFGS model (3.15) with rapid calculating ability for FDNO (3.1) is substantiated through numerical experiments in the presence of noises. Meanwhile, several representative existing models, which are DTND model (3.9), three-step DTND model (3.10), four-step DTND model (3.11), five-step DTND model (3.12), and NRI (3.13) model, are also used to solve the same FDNO problem in the presence of noises for comparison.

3.4.1 Example 1

In this subsection, one FDNO benchmark problem originated from [25] is addressed by the abovementioned discrete models:

$$\begin{aligned}
& \min_{\mathbf{y}(t_{k+1}) \in \mathbb{R}^4} \Phi(\mathbf{y}(t_k), t_k) \\
& = (y_1(t_k) + t_k)^2 + (y_2(t_k) + t_k)^2 + (y_3(t_k) - \\
& \quad \exp(-t_k))^2 + 0.1(t_k - 1)y_3(t_k)y_4(t_k) - (y_1(t_k) + \\
& \quad \ln(0.1t_k + 1))(y_2(t_k) + \sin(t_k)) + (y_1(t_k) + \\
& \quad \sin(t_k))y_3(t_k) + (y_4(t_k) + \exp(-t_k))^2,
\end{aligned} \tag{3.44}$$

where $[k\delta, (k+1)\delta] \in [0, 10]$ denotes the k th time computing interval, $\forall k \in \{0, 1, 2, \dots\}$, and $\mathbf{y}(t_k) = [y_1(t_k), y_2(t_k), y_3(t_k), y_4(t_k)]$. In order to evaluate different performances generated by models just mentioned, we choose the L_2 norm of residual error which is $\|\xi(t_k)\|_2 = \|\partial\Phi(\mathbf{y}(t), t)/\partial\mathbf{y}(t)\|_2$ as the monitoring variable.

To begin with, Fig. 3.3 shows comparative performances of NRI model (3.13) and NTND model (3.14) in the presence of random noises. In addition, the starting states $\mathbf{y}(t_0)$ used in the computing process is randomly generated. Specifically

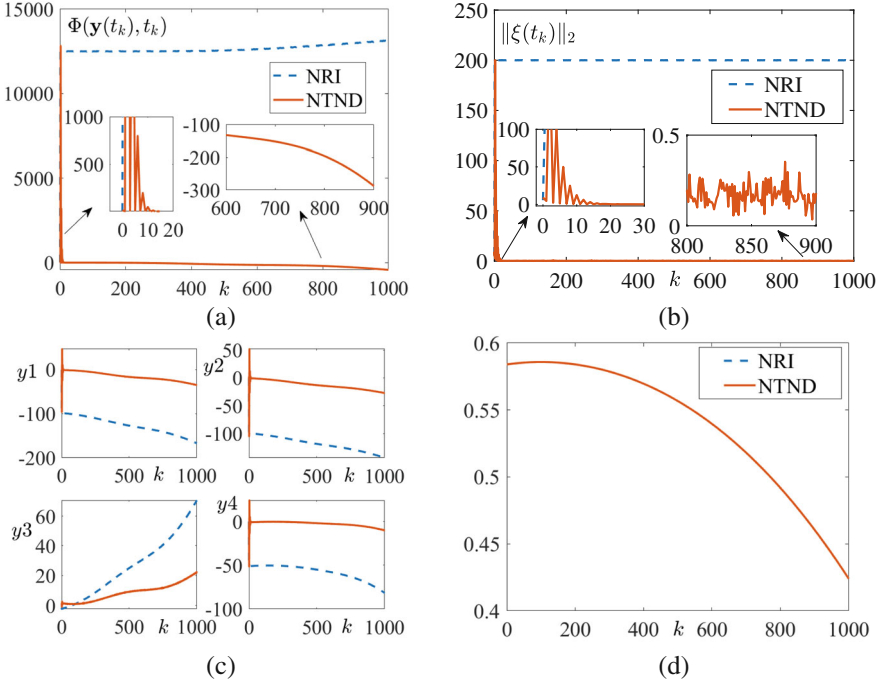


Fig. 3.3 Comparisons of NRI model (3.13) and NTND model (3.14) for solving FDNO (3.44) with $\delta = 0.01$ s and random noise $\rho \in [99.9, 100.1]$. (a) Comparison of $\Phi(\mathbf{y}(t_k), t_k)$, (b) $\|\xi(t_k)\|_2$ of NRI model (3.13) and NTND model (3.14), and (c) Comparison of $\mathbf{y}(t_k)$ (d) Comparison of minimal eigenvalue of Hessian matrix

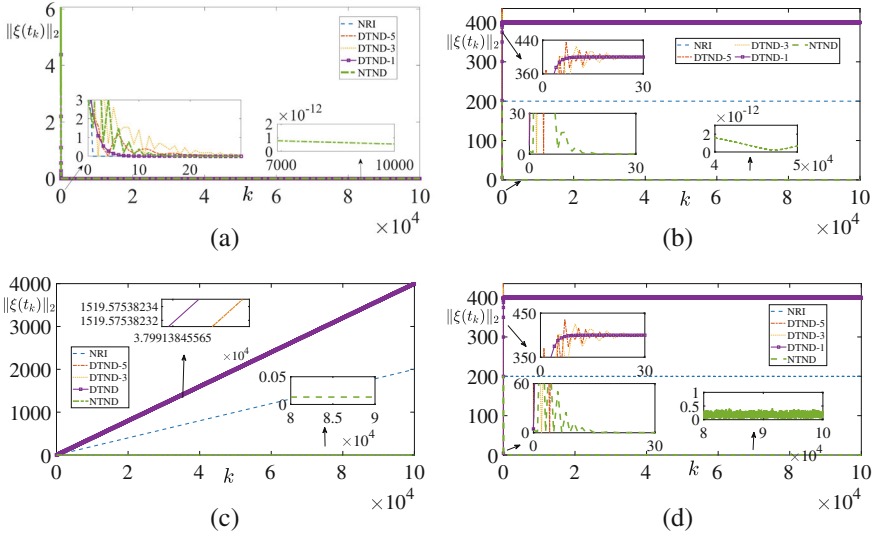


Fig. 3.4 Convergence performance of the constructed NTND model (3.14), NTND-BFGS model (3.15), DTND model (3.9), three-step DTND model (3.10), five-step DTND model (3.12), and NRI model (3.13) and for solving FDNO (3.44) at $\delta = 0.0001$ s (a) without noises, (b) constant noises, (c) linear noises, and (d) random noises

speaking, as demonstrated in Fig. 3.3a, NTND model (3.14) successfully obtains the minimum value of $\Phi(\mathbf{y}(t_k), t_k)$ at each time instant, while NRI model (3.13) fails to cope with the interference caused by random noises. Furthermore, Fig. 3.3b shows that the residual error of NTND model (3.14) converges to a very small value which is basically 10^3 times less than that of NRI model (3.13). In addition, the comparison of each element trajectory of $\mathbf{y}(t_k)$ is plotted in Fig. 3.3c. As shown in Fig. 3.3d, Hessian matrix minimal eigenvalues of FDNO benchmark problem (3.44) obtained from two models coincide with each other, which are both larger than zero during the computational time. That is the prerequisites for utilizing NRI model (3.13) and NTND model (3.14) on FDNO benchmark problem (3.44).

Next, no matter what kind of noises they are, we can use combinations of constant noises, linear noises, and random noises to define it mathematically. Thereby, it is allowed to explore effects which noises exert on different models in a categorical way. Experimental results of FDNO benchmark problem (3.44) among all the aforementioned models in the presence of different kinds of additive noises are plotted in Figs. 3.4, 3.5, and 3.6.

- (1) *No noises*: Observing the contrasting principle, Figs. 3.4a–3.6a display results of benchmark problem (3.44) as to no noises. All the discrete-time models perform accurately under the condition that there is no perturbation. Besides, NTND model (3.14) synthesizes the smallest residual error compared with other discrete-time existing models. From the perspective of convergence rate, NRI

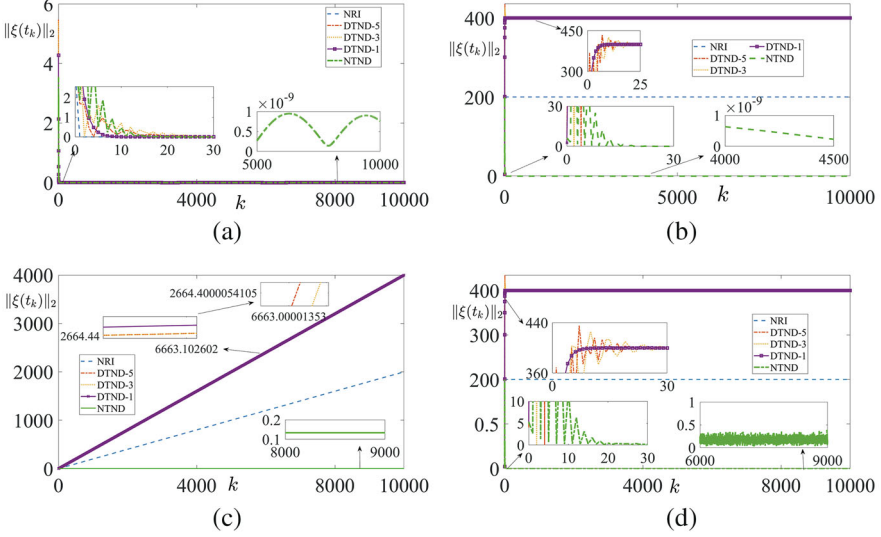


Fig. 3.5 Convergence performance of the designed NTND model (3.14), NTND-BFGS model (3.15), DTND model (3.9), three-step DTND model (3.10), five-step DTND model (3.12), and NRI model (3.13) and for solving FDNO (3.44) at $\delta = 0.001$ s (a) without noises, (b) constant noises, (c) linear noises, and (d) random noises

- model (3.13) has the best performance. Meanwhile, subgraphs of Figs. 3.4a–3.6a indicate that NTND model (3.14) converges a little bit slower than DTND model (3.9) and NRI model (3.13), and the approximating integration term inevitably causes extra updates.
- (2) *Constant noises:* As shown in Figs. 3.4b–3.6b, DTND model (3.9), three-step DTND model (3.10), five-step DTND model (3.12), and NRI model (3.13) basically fail to handle constant noises. The residual error generated by NRI model (3.13) is smaller than these three DTND models. Figures 3.4a–3.6a indicate that constant noises have negligible influence on existing models with residual error being about 2×10^2 or 4×10^2 . In contrast, since associating the error-integration term with the given NTND model (3.14) and the NTND-BFGS model (3.15), residual error values of those two are expedited from order 10^{-7} to 10^{-12} , with time sampling interval being 0.01, 0.001, and 0.0001 s. In addition, the convergence rate of NTND-BFGS model (3.15) is slightly more rapid than that of NTND model (3.14).
 - (3) *Linear time-varying noises:* As observed from Figs. 3.4c–3.6c, when linear time-varying noise $\varepsilon(t) = 100(t - 1)$, the residual error of NTND model (3.14) holds at the order of 10^0 , 10^{-1} , and 10^{-2} , with time sampling interval being reduced 10 times in turn. The above result of convergence rate verifies that NTND model (3.14) follows $\mathbf{O}(\delta)$ convergent speed proved in Theorem 3.3.

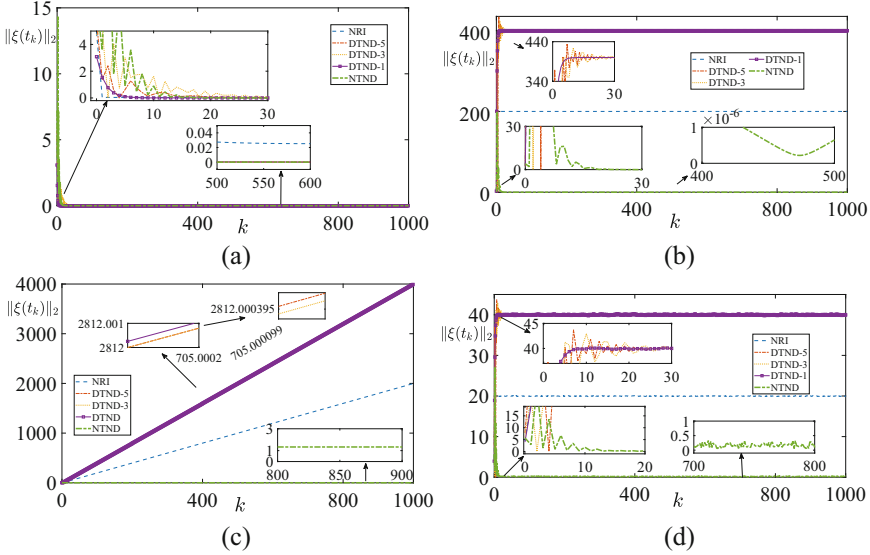


Fig. 3.6 Convergence performance of the given NTND model (3.14), NTND-BFGS model (3.15), DTND model (3.9), three-step DTND model (3.10), five-step DTND model (3.12), and NRI model (3.13) and for solving FDNO (3.44) at $\delta = 0.01$ s (a) without noises, (b) constant noises, (c) linear noises, and (d) random noises

For the existing models, with the presence of linear noises, the residual error linearly changes with time as a rather large slope. Therefore, DTND model (3.9), three-step DTND model (3.10), five-step DTND model (3.12), and NRI model (3.13) are not capable of solving FDNO problems with linear noises.

- (4) *Random noises*: From Figs. 3.4d–3.6d, it is evident that the maximum steady residual error synthesized by NTND model (3.14) is restricted to 0.2, which is 1×10^2 times less than the mean value of input random noises. To be specific, results of FDNO (3.44) basically remains unchanged in different sampling intervals, which means NTND model (3.14) maintains rather bounded residual error output as large random noises are inputted. It is also worth pointing out that the speed of NTND model (3.14) converging to near zero is rapid. In contrast, other existing models have pretty high residual error. These experimental results validate the superiority and applicability of the presented NTND model (3.14) for the FDNO problem, especially as to deal with strong random noises.

Table 3.1 is further given, which compares models of NTND model (3.14), NTND-BFGS model (3.15), and existing discrete models for solving FDNO problem (3.44) in the presence of noises. Summing up, we show how the theoretical analyses of NTND model (3.14) and NTND-BFGS model (3.15) are substantiated by experimental results.

Table 3.1 Comparison between methods for solving future dynamic nonlinear optimization problem (3.44)

Method	Residual error in terms of different kinds of noises				Property in noisy environment		Essentially designed to solve time-variant problems
	Constant noises	Linear noises	Random noises	Zero noises	Convergent	Noise-tolerant	
NTND (3.14)	2.9×10^{-13}	1.0×10^{-2}	2.5×10^{-1}	7.6×10^{-13}	Yes	Yes	Yes
NTND-BFGS (3.15)	7.2×10^{-13}	3.3×10^{-2}	1.8×10^{-1}	1.4×10^{-12}	Yes	Yes	Yes
One-step DTND (3.9)	8.0×10^2	Infinite	4.0×10^2	1.6×10^{-8}	No†	No	Yes
Three-step DTND (3.10)	8.0×10^2	Infinite	4.0×10^2	2.7×10^{-8}	No†	No	Yes
Four-step DTND (3.11)	8.0×10^2	Infinite	4.0×10^2	7.7×10^{-8}	No†	No	Yes
Five-step DTND (3.12)	8.0×10^2	Infinite	4.0×10^2	2.7×10^{-8}	No†	No	Yes
NRI model (3.13)	4.0×10^2	Infinite	2.0×10^2	4.5×10^{-4}	No†	No	No

Parameters involved are set as follows: $\delta = 0.0001$ s, $c_2 = 0.6$, constant noise $\varepsilon(t) = 200$, linear noise $\varepsilon(t) = 100(t - 1)$, random noise $\varepsilon(t) \in [99.9, 100.1]$

For c_1 , four-step DTND (3.11) is 0.05; the other aforementioned models are 0.5

No† means that influences exerted by noises are nonnegligible

Overall, FDNO benchmark problem (3.44) verifies that the constructed NTND model (3.14) and NTND-BFGS model (3.15) possess extraordinary ability in suppressing different kinds of noises, even their combinations, without any prior information of noises.

3.5 Application to Motion Generation

Robotic systems have evolved rapidly in engineering fields [38–41]. In this section, we apply NTND model (3.14), NTND-BFGS model (3.15), and NRI model (3.13) to solve the inverse kinematics of a two-link planar robot. Afterwards, let $\mathbf{a}(t) \in \mathbb{R}^2$ denote the vector of practical end effector position in the Cartesian coordinate system, meanwhile $\mathbf{a}_d(t)$ is the corresponding desired one; $\boldsymbol{\theta}(t) = [\theta_1(t), \theta_2(t)]^T$ symbolizes the joint-angle vector; $f(\cdot)$ represents the forward kinematics mapping relation between the end effector position and the joint angle [40], which means $f(\boldsymbol{\theta}(t)) = \mathbf{a}(t)$. Considering T as the end point of the solving process, every computational time interval can be expressed by $[k\delta, (k+1)\delta) \in [0, T]$, where $k = 0, 1, 2, \dots$ and δ is the sampling gap. Thereby, the cost function of the aforementioned robot motion could be defined as

$$\Phi(\boldsymbol{\theta}_k, t_k) = f^T(\boldsymbol{\theta}_k)f(\boldsymbol{\theta}_k) - 2\mathbf{a}_d^T(t_k)f(\boldsymbol{\theta}_k). \quad (3.45)$$

Evidently, the solution of $\mathbf{a}(t)$ tracking $\mathbf{a}_d(t)$ can be classified as the following FDNO problem:

$$\min_{\boldsymbol{\theta}_{k+1} \in \mathbb{R}^2} \Phi(\boldsymbol{\theta}_{k+1}, t_{k+1}). \quad (3.46)$$

For simple illustration, each link length is set as 1 m. To be specific, our aim is to use the two-link planar robot manipulator with NTND model (3.14) and NRI model (3.13) embedded to draw a four-leaf clover.

The ensuring figures show comparative experimental results conducted under various experimental environments which differs in kinds of additive noises. For comparison, Fig. 3.7 plots profiles of the whole tracking motion trajectories, joint angle, and the position error without the presence of noises. Through Figs. 3.8, 3.9, and 3.10, it is shown that the given motion is completed well by NTND model (3.14) where the position error is slight, despite additive noises. Specifically, adding constant noise $\varepsilon(t) = 10$ to joint velocities when being solved by NTND model (3.14), the position error of the manipulator's end effector in Fig. 3.8a is less than 10^{-9} m. Thus, the designed NTND model (3.14) is feasible for industrial applications.

Fig. 3.7 End effector movement of a two-link robot following a four-leaf-clover trajectory synthesized by NTND model (3.14) with $\delta = 0.001$ s. **(a)** Motion trajectories without perturbation, **(b)** position errors of experimental joint trajectories, and **(c)** profiles of position error in the Cartesian coordinate system

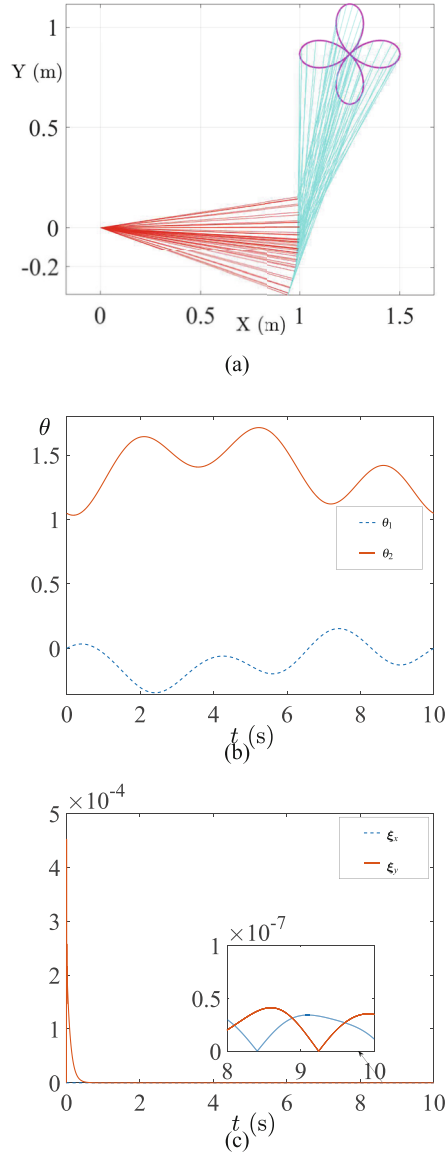


Fig. 3.8 Position errors in the Cartesian coordinate system of a two-link robot tracking an ideal four-leaf-clover path obtained from NTND model (3.14) with constant noise $\varepsilon(t) = 10$. **(a)** $\delta = 0.0001$ s, **(b)** $\delta = 0.001$ s, and **(c)** $\delta = 0.01$ s

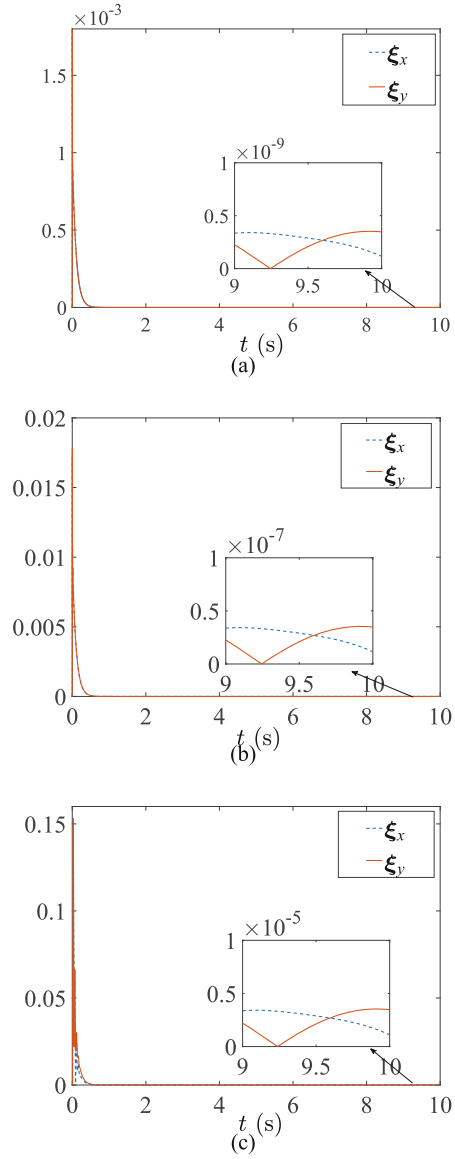


Fig. 3.9 Position errors in the Cartesian coordinate system of a two-link robot tracking an ideal four-leaf-clover path obtained from NTND model (3.14) with linearly time-dependent noise $\varepsilon(t) = 10(t - 1)$. **(a)** $\delta = 0.0001$ s, **(b)** $\delta = 0.001$ s, and **(c)** $\delta = 0.01$ s

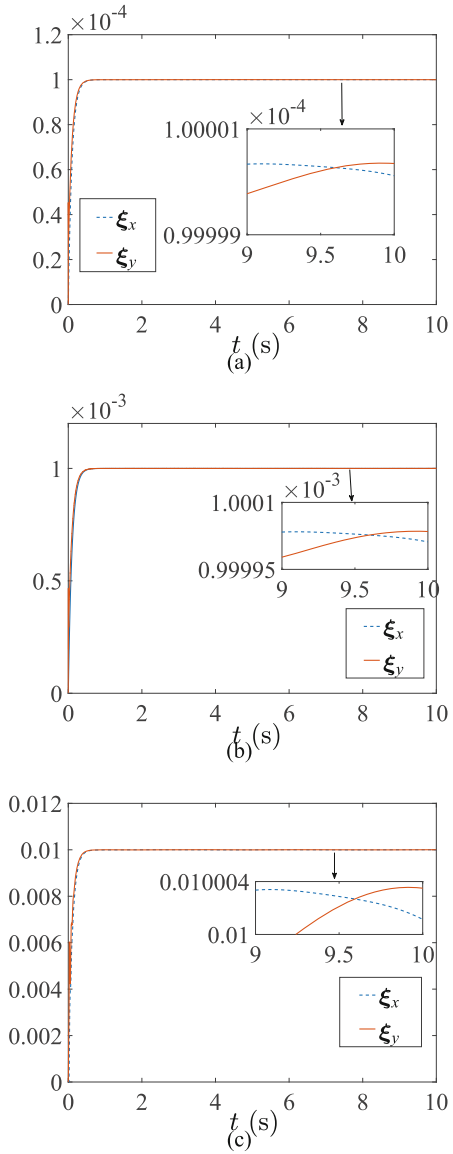
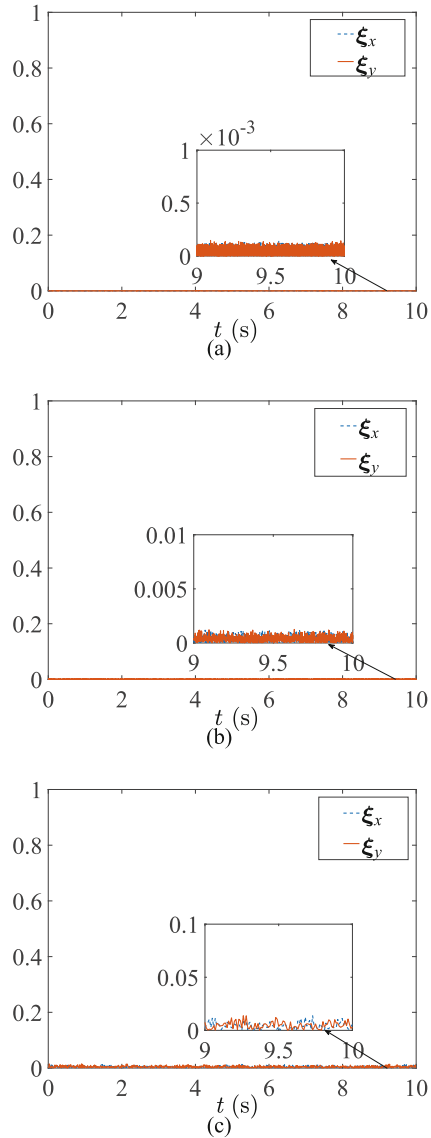


Fig. 3.10 Position errors in the Cartesian coordinate system of a two-link robot tracking an ideal four-leaf-clover path obtained from NTND model (3.14) with random noise $\varepsilon(t) \in [9, 11]$. (a) $\delta = 0.0001$ s, (b) $\delta = 0.001$ s, and (c) $\delta = 0.01$ s



3.6 Conclusions

In this chapter, NTND model (3.14) has been addressed for future dynamic nonlinear optimization problems in the presence of a class of noises affecting the system. Integration control from control technique has been defined to minimize the cost function rapidly. The use of the quasi-Newton BFGS method has been

given through NTND-BFGS model (3.15), which cannot only deal with noises for the FDNO problem, but eliminate the expensive calculation of inversion matrices. Results obtained by numerical experiments are reported in comparison with DTND model (3.9), three-step DTND model (3.10), five-step DTND model (3.12), and NRI model (3.13), thereby highlighting the superiority of the presented models in robustness, efficacy, and computational complexity when solving the FDNO problem with noises.

References

1. Zhang Y, Wang J (2002) A dual neural network for constrained joint torque optimization of kinematically redundant manipulators. *IEEE Trans Syst Man Cybern-Part B Cybern* 32(5):654–662
2. Grosan C, Abraham A (2008) A new approach for solving nonlinear equations systems. *IEEE Trans Syst Man Cybern-Part A Syst Hum* 38(3):698–714
3. Yang C, Huang K, Cheng H, Li Y, Su CY (2017) Haptic identification by ELM-controlled uncertain manipulator. *IEEE Trans Syst Man Cybern Syst* 47(8):2398–2409
4. Liu YJ, Li S, Tong S, Chen CP (2016) Neural approximation-based adaptive control for a class of nonlinear nonstrict feedback discrete-time systems. *IEEE Trans Neural Netw Learn Syst* 28(7):1531–1541
5. Liu YJ, Tong S, Chen CP, Li DJ (2016) Adaptive NN control using integral Barrier Lyapunov functionals for uncertain nonlinear block-triangular constraint systems. *IEEE Trans Cybern* 47(11):3747–3757
6. Guo D, Nie Z, Yan L (2017) Novel discrete-time Zhang neural network for time-varying matrix inversion. *IEEE Trans Syst Man Cybern Syst* 47(8):2301–2310
7. Liu YJ, Lu S, Tong S, Chen X, Chen CP, Li DJ (2018) Adaptive control-based Barrier Lyapunov functions for a class of stochastic nonlinear systems with full state constraints. *Automatica* 87:83–93
8. Liu YJ, Zeng Q, Liu L, Tong S (2018) An adaptive neural network controller for active suspension systems with hydraulic actuator. *IEEE Trans Syst Man Cybern Syst* 50(12):5351–5360
9. Yue L, Yang Y (2013) A new integral filter algorithm for unconstrained global optimization. *Numer Algorithms* 63:419–430
10. Hager WW, Zhang H (2006) A survey of nonlinear conjugate gradient methods. *Pac J Optim* 2(1):35–58
11. Andrei N (2014) An accelerated subspace minimization three-term conjugate gradient algorithm for unconstrained optimization. *Numer Algorithms* 65:859–874
12. Liufu Y, Jin L, Shang M, Wang X, Wang, FY (2024) ACP-incorporated perturbation-resistant neural dynamics controller for autonomous vehicles. *IEEE Trans Intell Veh*. In Press. <https://doi.org/10.1109/TIV.2023.3348632>
13. Yan J, Liu M, Jin L (2023) Cerebellum-inspired model predictive control for redundant manipulators with unknown structure information. *IEEE Trans Cogn Dev Syst*. In Press. <https://doi.org/10.1109/TCDS.2023.3340179>
14. Jin L, Li S (2016) Distributed task allocation of multiple robots: a control perspective. *IEEE Trans Syst Man Cybern Syst* 48(5):693–701
15. Yang C, Jiang Y, He W, Na J, Li Z, Xu B (2018) Adaptive parameter estimation and control design for robot manipulators with finite-time convergence. *IEEE Trans Ind Electron* 65(10):8112–8123

16. Cui R, Yang C, Li Y, Sharma S (2017) Adaptive neural network control of AUVs with control input nonlinearities using reinforcement learning. *IEEE Trans Syst Man Cybern Syst* 47(6):1019–1029
17. Liu YJ, Tong S (2016) Optimal control-based adaptive NN design for a class of nonlinear discrete-time block-triangular systems. *IEEE Trans Cybern* 46(11):2670–2680
18. Liu YJ, Li J, Tong S, Chen CP (2016) Neural network control based adaptive learning design for nonlinear systems with full state constraints. *IEEE Trans Neural Netw Learn Syst* 27(7):1562–1571
19. He W, Dong Y, Sun C (2015) Adaptive neural impedance control of a robotic manipulator with input saturation. *IEEE Trans Syst Man Cybern Syst* 46(3):334–344
20. Xiao L, Lu R (2015) Finite-time solution to nonlinear equation using recurrent neural dynamics with a specially-constructed activation function. *Neurocomputing* 151:246–251
21. Jin L, Li S, Xiao L, Lu R, Liao B (2017) Cooperative motion generation in a distributed network of redundant robot manipulators with noises. *IEEE Trans Syst Man Cybern Syst* 48(10):1715–1724
22. Xiao L, Li S, Li K, Jin L, Liao B (2018) Co-design of finite-time convergence and noise suppression: a unified neural model for time varying linear equations with robotic applications. *IEEE Trans Syst Man Cybern Syst* 50(12):5233–5243
23. Zhang Y, Zhang Y, Chen D, Xiao Z, Yan X (2016) From Davidenko method to Zhang dynamics for nonlinear equation systems solving. *IEEE Trans Syst Man Cybern Syst* 47(11):2817–2830
24. Jin L, Li S, Hu B, Liu M, Yu J (2018) Noise-suppressing neural algorithm for solving time-varying system of linear equations: a control-based approach. *IEEE Trans Ind Inform* 15(1):236–246
25. Jin L, Zhang Y (2016) Continuous and discrete Zhang dynamics for real-time varying nonlinear optimization. *Numer Algorithms* 73:115–140
26. Liu YJ, Tong S, Chen CP, Li DJ (2015) Neural controller design-based adaptive control for nonlinear MIMO systems with unknown hysteresis inputs. *IEEE Trans Cybern* 46(1):9–19
27. Chen X, Jin L, Li S (2024) An online learning strategy for echo state network. *IEEE Trans Syst Man Cybern Syst* 54(1):644–655
28. Selvan SE, Amato U, Gallivan KA, Qi C, Carfora MF, Larobina M, Alfano B (2012) Descent algorithms on oblique manifold for source-adaptive ICA contrast. *IEEE Trans Neural Netw Learn Syst* 23(12):1930–1947
29. Tivive FHC, Bouzerdoum A (2005) Efficient training algorithms for a class of shunting inhibitory convolutional neural networks. *IEEE Trans Neural Netw* 16(3):541–556
30. Li DH, Fukushima M (2001) A modified BFGS method and its global convergence in nonconvex minimization. *J Comput Appl Math* 129(1–2):15–35
31. Jin L, Zhang Y (2014) Discrete-time Zhang neural network for online time-varying nonlinear optimization with application to manipulator motion generation. *IEEE Trans Neural Netw Learn Syst* 26(7):1525–1531
32. Zhang Y, Ma W, Cai B (2008) From Zhang neural network to Newton iteration for matrix inversion. *IEEE Trans Circuits Syst I: Reg Pap* 56(7):1405–1415
33. Zhang Y, He L, Hu C, Guo J, Li J, Shi Y (2019) General four-step discrete-time zeroing and derivative dynamics applied to time-varying nonlinear optimization. *J Comput Appl Math* 347:314–329
34. Qiu B, Zhang Y, Guo J, Yang Z, Li X (2019) New five-step DTZD algorithm for future nonlinear minimization with quartic steady-state error pattern. *Numer Algorithms* 81:1043–1065
35. Liu D, Li H, Wang D (2014) Online synchronous approximate optimal learning algorithm for multi-player non-zero-sum games with unknown dynamics. *IEEE Trans Syst Man Cybern Syst* 44(8):1015–1027
36. Hu JL, Wu Z, McCann H, Davis LE, Xie CG (2006) BFGS quasi-Newton method for solving electromagnetic inverse problems. *IEE Proc-Microwaves Antennas Propag* 153(2):199–204
37. Weingarten H, Steinberg Y, Shamai SS (2006) The capacity region of the Gaussian multiple-input multiple-output broadcast channel. *IEEE Trans Inf Theory* 52(9):3936–3964

38. Yang C, Luo J, Pan Y, Liu Z, Su CY Personalized variable gain control with tremor attenuation for robot teleoperation. *IEEE Trans Syst Man Cybern Syst* 48(10):1759–1770
39. Yang C, Wang X, Li Z, Li Y, Su CY (2016) Teleoperation control based on combination of wave variable and neural networks. *IEEE Trans Syst Man Cybern Syst* 47(8):2125–2136
40. 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
41. Liu M, Hu Y, Jin L (2024) Discrete data-driven control of redundant manipulators with adaptive jacobian matrix. *IEEE Trans Ind Electron. In Press.* <https://doi.org/10.1109/TIE.2023.3347831>

Chapter 4

Computational Neural Dynamics



4.1 Introduction

Nonlinear optimization is considered to be a vital issue in the science and engineering fields due to its wide applications [1–9], which include robotics [1, 2], winner-take-all (WTA) operation [3, 4], multi-agent systems [5, 6], and many other technologies [7–9]. Thus, it attracts considerable attention of researchers, and a growing number of publications on such problems are appearing. Specifically, the Newton iteration algorithm and gradient-related methods are popular in these studies [10–14]. For example, Nazemi et al. construct a gradient-based neural network method in [11] for the solution of strictly convex quadratic programming (QP) problems. In [12], a gradient-based two-stage optimization strategy is constructed by Chai et al. and applied to the autonomous vehicle parking motion planning. In [13], Li et al. develop a modified BFGS method for unconstrained optimization, which is confirmed to be globally convergent. In spite of the extraordinary progress made in optimization problem-solving, one important challenge to note is that these reported studies are mostly investigated under the condition that the parameters of the optimization problems are time-invariant. The lack of time-derivative information of time-variant parameters makes it challenging for the static solvers to track the dynamic theoretical solution of the time-variant problems in a predictive way, which leads to unavoidable lagging errors [15, 16]. Thus, these methods might not be suitable for time-variant systems.

Being characterized as convenient for hardware implementation and parallel computing, neural networks and the related neural dynamics approaches become a particularly important method for real-time optimization problems [17–19]. Subsequently, various neural network methods are explored to solve time-variant optimization problems, and the recurrent neural network (RNN) is a promising one [20–22]. Among various RNN-based methods, zeroing neural network (ZNN) is currently regarded as one of the most efficient methods to tackle time-variant computational problems, such as time-variant complex linear matrix equations [23],

time-variant optimization [24], and Sylvester equation [25]. By utilizing the time-derivative information, lagging error is eliminated, and many real-time nonlinear problems get their accurate solutions. It is shown in [26] and [27] that different continuous ZNN models are established aiming at time-variant QP problems with equality and inequality constraints, which are converted into nonlinear equations with their solutions corresponding to the equilibrium points of the neural networks. In addition to these continuous ZNN models, some discrete neural dynamics models are also employed to solve time-varying optimization problems [28, 29]. For example, two discrete-time computing models are constructed in [29] by employing two different types of Taylor-type difference rules, which successfully tackle time-variant nonlinear minimization without constraints with the steady-state residual errors varying in an $O(\theta^3)$ manner, where θ is the sampling gap. These methods have been proved to be advantageous in some situations, but generally, they are not entirely feasible, and there remain a number of important aspects for further investigation and improvement.

That time steps and sampling gap vary with time makes the continuous-time system difficult to be implemented on digital computers directly [29]. Discretization is a necessary and effective approach to cope with this challenge, and the Euler forward-difference rule is often applied for this work [30, 31]. Equality and inequality constraints of optimization problems are indispensable in many practical applications. For example, the motion tracking task of redundant robots requires the equality constraints, while in WTA operation, both equality constraints and inequality constraints are needed. Considering that the existing discrete-time neural dynamics methods fail to deal with inequality constraints in nonlinear optimization, the time-variant nonlinear optimization with equality and inequality constraints (TVNOEIC) is transformed into a time-variant nonlinear equation with the same solution as the former. In addition, most existing models only focus on the case that the whole solving process is free of noise interference, which is not in line with practice. If the noise is not properly handled, the precision will considerably decrease. The previous studies indicate that the summation term in the model is helpful for resisting noise, and the error summation term is thus considered to design the model. All things considered, a noise-suppressing discrete-time neural dynamics (NSDTND) model that can solve TVNOEIC problems is presented in this chapter. To illustrate the performance of the given model, comprehensive comparisons among the existing methods for nonlinear optimization are summarized in Table 4.1. As seen from this table, the designed NSDTND model performs well as compared with existing methods. Specifically, this chapter studies a general nonlinear objective function rather than a quadratic one as presented in [2, 26, 27]. Besides, different from conventional static methods [12, 13], this chapter investigates time-variant problems, which is more challenging. What is more, the inequality constraint in the time-variant nonlinear optimization has not been handled in a discrete-time manner yet [14, 16, 29], and this is the first time that the optimization problem

Table 4.1 Comparisons among different methods for time-variant nonlinear optimization

Method	Dynamic vs. static	Problem type	Equality constraint	Inequality constraint	Noise tolerance	Easy for hardware implementation
Model (4.10)	Dynamic	NO [†]	Yes	Yes	Yes	Yes
NI (4.11)	Static	NO [†]	Yes	Yes	No	Yes
MNI (4.12)	Dynamic	NO [†]	No	No	Yes	Yes
[2]	Dynamic	QP	Yes	No	Yes	No
[3, 11]	Static	QP	Yes	Yes	No	No
[12, 13]	Static	NO [†]	No	No	No	Yes
[16]	Dynamic	Nonlinear optimization	Yes	No	No	Yes
[26, 27]	Dynamic	QP	Yes	Yes	No	No
[29]	Dynamic	NO [†]	No	No	No	Yes

[†] — NO means nonlinear optimization

with inequality constraint is successfully solved on the premise of guaranteeing easy hardware implementation (i.e., discrete-form) and noise suppression.

This chapter is organized in the following way. In Sect. 4.2, the problem formulation of TVNOEIC is provided, and then the design process, as well as the discretization process of NSDTND model for solving TVNOEIC problems, is elaborated. Then, mathematical and theoretical analyses are presented in Sect. 4.3 to prove the convergence property of the constructed model. After this, Sect. 4.4 provides numerical simulations and shows the WTA application. Finally, Sect. 4.5 presents a brief review of the whole work and identifies directions for further research. The main contributions of this chapter are as follows:

- This chapter proposes an NSDTND model with noise-suppressing ability, which not only solves time-variant optimization problems but also handles equality and inequality constraints in a discrete-time manner.
- Theoretical analyses are provided and numerical experiments are conducted, which guarantee that the presented NSDTND model performs well on TVNOEIC problem-solving and shows superior performance over the SOTA methods.
- The NSDTND model is successfully applied to the WTA operation, and its effectiveness and feasibility in practice are verified.

4.2 Problem Formulation and NSDTND Model

In this section, the TVNOEIC problem is formulated. Thereafter, a continuous model originating from the design ideas of ZNN is presented. Then, the continuous model is discretized into the NSDTND model for easier hardware implementation.

4.2.1 Problem Formulation

A general TVNOEIC problem is depicted as

$$\begin{aligned} \min & f(\mathbf{x}(t), t), \\ \text{s. t. } & \mathbf{u}(\mathbf{x}(t)) = \mathbf{a}(t) + B(t)\mathbf{x}(t) = \mathbf{0}, \\ & \mathbf{v}(\mathbf{x}(t)) = \mathbf{c}(t) + D(t)\mathbf{x}(t) \leq \mathbf{0}, \end{aligned} \quad (4.1)$$

where $t \in [0, +\infty)$ is defined as time instant; the term $\mathbf{x}(t) \in \mathbb{R}^m$ refers to the unknown vector to be determined; $f(\cdot) : \mathbb{R}^m \rightarrow \mathbb{R}$ denotes the nonlinear objective function, which is limited to be continuously differentiable and convex on $\mathbf{x}(t)$ at any time t to ensure that the solution is globally optimal rather than locally optimal; functions $\mathbf{u}(\cdot) : \mathbb{R}^m \rightarrow \mathbb{R}^n$ and $\mathbf{v}(\cdot) : \mathbb{R}^m \rightarrow \mathbb{R}^p$ are equality constraint and inequality constraint, respectively; vector $\mathbf{a}(t) \in \mathbb{R}^n$, vector $\mathbf{c}(t) \in \mathbb{R}^p$, full row rank matrix $B(t) \in \mathbb{R}^{n \times m}$ and matrix $D(t) \in \mathbb{R}^{p \times m}$ indicate four time-variant parameters, respectively.

For ease of expression, the gradient of function $f(\mathbf{x}(t), t)$ with respect to $\mathbf{x}(t)$ is defined as

$$\mathbf{h}(\mathbf{x}(t), t) = \frac{\partial f(\mathbf{x}(t), t)}{\partial \mathbf{x}(t)} = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_m} \right]^T, \quad (4.2)$$

where symbol T is the transpose of the original matrix or vector. In order to obtain the feasible solution to the TVNOEIC problem (4.1), it is transformed into a nonlinear system of equations, as illustrated in [27].

Lemma 1 ([27]) *The optimal solution to the TVNOEIC problem (4.1) is obtained if and only if vectors $\boldsymbol{\rho}(t) \in \mathbb{R}^n$, $\boldsymbol{\sigma}(t) \in \mathbb{R}^p$ satisfy*

$$\begin{cases} \partial \mathbf{h}(\mathbf{x}(t), t) + B^T(t)\boldsymbol{\rho}(t) + D^T(t)\boldsymbol{\sigma}(t) = \mathbf{0}, \\ \mathbf{a}(t) + B(t)\mathbf{x}(t) = \mathbf{0}, \\ \psi_{FB}^v(-\mathbf{c}(t) - D(t)\mathbf{x}(t), \boldsymbol{\sigma}(t)) = \mathbf{0}, \end{cases} \quad (4.3)$$

where $\boldsymbol{\rho}(t) = [\rho_1(t), \rho_2(t), \dots, \rho_n(t)]^T$ and $\boldsymbol{\sigma}(t) = [\sigma_1(t), \sigma_2(t), \dots, \sigma_p(t)]^T$ separately indicate the Lagrangian multipliers of the equality and inequality constraints. Given $\mathbf{v} > \mathbf{0} \in \mathbb{R}^p$, function $\psi_{FB}^v(\mathbf{v}, \mathbf{r}, \mathbf{s})$ is defined as

$$\psi_{FB}^v(\mathbf{v}, \mathbf{r}, \mathbf{s}) = (\mathbf{r} + \mathbf{s}) - \sqrt{\mathbf{r} \circ \mathbf{r} + \mathbf{s} \circ \mathbf{s} + \mathbf{v}}, \quad (4.4)$$

where $\mathbf{r} \in \mathbb{R}^p$ and $\mathbf{s} \in \mathbb{R}^p$ are vectors whose element values are greater than zero, and symbol $\circ$ indicates the Hadamard product.

Similar to [27], formula (4.3) is further reformulated as the following matrix equation to facilitate the solvability:

$$G(t)y(t) + w(t) = \mathbf{0}, \quad (4.5)$$

where

$$G(t) = \begin{bmatrix} \mathbf{0} & B^T(t) & D^T(t) \\ B(t) & \mathbf{0} & \mathbf{0} \\ -D(t) & \mathbf{0} & \mathbf{I} \end{bmatrix} \in \mathbb{R}^{(m+n+p) \times (m+n+p)}, \quad y(t) = \begin{bmatrix} x(t) \\ \rho(t) \\ \sigma(t) \end{bmatrix} \in \mathbb{R}^{m+n+p},$$

$$w(t) = \begin{bmatrix} h(x(t), t) \\ a(t) \\ w_3(t) \end{bmatrix} \in \mathbb{R}^{m+n+p}, \quad w_3(t) = -c(t) - M(t) \in \mathbb{R}^p,$$

$$M(t) = \sqrt{\delta(t) \circ \delta(t) + \sigma(t) \circ \sigma(t) + v} \in \mathbb{R}^p, \quad \delta(t) = -c(t) - D(t)x(t) \in \mathbb{R}^p,$$

with $\mathbf{I}$ denoting the identity matrix.

4.2.2 Continuous-Time Neural Dynamics Model

Following the design idea of the ZNN method, we should at first define an error function to formulate Eq. (4.5) into a zero-finding problem as follows:

$$\varepsilon(t) = G(t)y(t) + w(t). \quad (4.6)$$

An integration-enhanced continuous-time neural dynamics algorithm $\dot{\varepsilon}(t) = -\alpha \varepsilon(t) - \beta \int_0^t \varepsilon(\tau) d\tau$ is exploited such that the error function can be forced to zero, where $\alpha > 0$ and $\beta > 0$ are scalar parameters. Updating the design formula with the error function (4.6) generates

$$\begin{aligned} \dot{G}(t)y(t) + G(t)\dot{y}(t) + \dot{w}(t) &= -\alpha(G(t)y(t) + w(t)) \\ &\quad - \beta \int_0^t (G(\tau)y(\tau) + w(\tau)) d\tau. \end{aligned} \quad (4.7)$$

4.2.3 NSDTND Model

Due to the fact that the above continuous-time neural dynamics model cannot be implemented on the digital computer directly, the Euler forward-difference formula is applied for discretizing the continuous model (4.7), which is shown as

$$\dot{x}_k = \frac{x_{k+1} - x_k}{\theta}, \quad (4.8)$$

where $\theta > 0$ denotes the sampling gap; $k = 0, 1, 2, \dots$ is the updating index; $\mathbf{x}_k$ indicates the abbreviation of $\mathbf{x}(t_k)$ with $t_k = k\theta$.

Then, we use formula (4.8) to approximate $\dot{\mathbf{y}}_k$. Note that in addition to $\dot{\mathbf{y}}_k$, model (4.7) also involves two derivative terms $\dot{G}$ and $\dot{\mathbf{w}}$, which are easily affected by noises when implemented in the circuit. For this reason, the Euler backward-difference rule is utilized to estimate these two parameters. Therefore, the continuous model (4.7) is discretized as

$$\begin{aligned} \mathbf{y}_{k+1} = & \mathbf{y}_k + G_k^\dagger(-(\mathbf{w}_k - \mathbf{w}_{k-1}) - (G_k - G_{k-1})\mathbf{y}_k \\ & - l_1(G_k\mathbf{y}_k + \mathbf{w}_k) - l_2 \sum_{i=0}^k (G_i\mathbf{y}_i + \mathbf{w}_i)), \end{aligned} \quad (4.9)$$

where † denotes the pseudo-inverse operation of a matrix, $l_1 = \alpha\theta$, and $l_2 = \beta\theta^2$. If the noise $\boldsymbol{\xi}(t) \in \mathbb{R}^{m+n+p}$ is considered, NSDTND model (4.9) is restructured as

$$\begin{aligned} \mathbf{y}_{k+1} = & \mathbf{y}_k + G_k^\dagger(-(\mathbf{w}_k - \mathbf{w}_{k-1}) - (G_k - G_{k-1})\mathbf{y}_k \\ & - l_1(G_k\mathbf{y}_k + \mathbf{w}_k) - l_2 \sum_{i=0}^k (G_i\mathbf{y}_i + \mathbf{w}_i) + \boldsymbol{\xi}(t_k)). \end{aligned} \quad (4.10)$$

4.2.4 Models for Comparison

For the purpose of comparison, Newton iteration (NI) for solving TVNOEIC problem (4.1) is generalized and formulated as

$$\mathbf{y}_{k+1} = \mathbf{y}_k + G_k^\dagger(-G_k\mathbf{y}_k - \mathbf{w}_k) = -G_k^\dagger\mathbf{w}_k. \quad (4.11)$$

Besides, a modified Newton integration (MNI) model for time-variant nonlinear optimization established in [14] is also presented for comparison, i.e.,

$$\mathbf{y}_{k+1} = \mathbf{y}_k + H^{-1}(\mathbf{x}_k, t_k)(-\mathbf{h}(\mathbf{x}_k, t_k) - \rho \sum_{i=0}^k \mathbf{h}(\mathbf{x}_i, t_i)), \quad (4.12)$$

where $H^{-1}(\mathbf{x}_k, t_k)$ indicates the inversion of Hessian matrix $H(\mathbf{x}_k, t_k) = \partial \mathbf{h}(\mathbf{x}_k, t_k) / \partial \mathbf{f}^T(\mathbf{x}_k, t_k) \in \mathbb{R}^{m \times m}$ and $\rho > 0$ is a scalar parameter.

4.3 Theoretical Analysis and Results

This section mainly includes theoretical analyses on the convergence and robustness of the given NSDTND model (4.9) for solving the TVNOEIC problem.

4.3.1 Precondition

To lay a groundwork for the proof on the convergence and stability, formula (4.9) is transformed into a concise form, as illustrated in the following theorem.

Theorem 4.1 *NSDTND model (4.9) is equivalent to*

$$(1 + l_1)\boldsymbol{\varepsilon}_k - \boldsymbol{\varepsilon}_{k-1} + l_2 \sum_{i=0}^k \boldsymbol{\varepsilon}_i + \mathbf{O}(\theta^2) = 0, \quad (4.13)$$

where $\mathbf{O}(\theta^2)$ is the truncation error vector with entries of order $O(\theta^2)$.

Proof Firstly, NSDTND model (4.9) is rearranged in the following form:

$$\begin{aligned} G_k(\mathbf{y}_{k+1} - \mathbf{y}_k) = & -(\mathbf{w}_k - \mathbf{w}_{k-1}) - (G_k - G_{k-1})\mathbf{y}_k \\ & - l_1\boldsymbol{\varepsilon}_k - l_2 \sum_{i=0}^k \boldsymbol{\varepsilon}_i. \end{aligned} \quad (4.14)$$

Making a simple transformation on Eq. (4.14) leads to

$$G_k \mathbf{y}_{k+1} = -(\mathbf{w}_k - \mathbf{w}_{k-1}) + G_{k-1} \mathbf{y}_k - l_1 \boldsymbol{\varepsilon}_k - l_2 \sum_{i=0}^k \boldsymbol{\varepsilon}_i. \quad (4.15)$$

According to the Taylor expansion [31], we get

$$\mathbf{y}_{k+1} = \mathbf{y}_k + \theta \dot{\mathbf{y}}_k + \mathbf{O}(\theta^2) \quad (4.16)$$

and

$$\mathbf{y}_{k-1} = \mathbf{y}_k - \theta \dot{\mathbf{y}}_k + \mathbf{O}(\theta^2). \quad (4.17)$$

Formula (4.15) is updated by (4.16) and (4.17), so that one can obtain

$$\begin{aligned} G_k(\mathbf{y}_k + \theta \dot{\mathbf{y}}_k + \mathbf{O}(\theta^2)) = & -(\mathbf{w}_k - \mathbf{w}_{k-1}) \\ & + G_{k-1}(\mathbf{y}_{k-1} + \theta \dot{\mathbf{y}}_k + \mathbf{O}(\theta^2)) \\ & - l_1 \boldsymbol{\varepsilon}_k - l_2 \sum_{i=0}^k \boldsymbol{\varepsilon}_i. \end{aligned} \quad (4.18)$$

Simplifying (4.18) generates

$$\begin{aligned} G_k \mathbf{y}_k + \mathbf{w}_k + \theta G_k \dot{\mathbf{y}}_k + \mathcal{O}(\theta^2) = & G_{k-1} \mathbf{y}_{k-1} + \mathbf{w}_{k-1} - l_1 \boldsymbol{\varepsilon}_k \\ & + \theta G_{k-1} \dot{\mathbf{y}}_k - l_2 \sum_{i=0}^k \boldsymbol{\varepsilon}_i, \end{aligned} \quad (4.19)$$

which can be further rewritten as the following equation by considering the error function (4.6):

$$\boldsymbol{\varepsilon}_k + \theta G_k \dot{\mathbf{y}}_k + \mathcal{O}(\theta^2) = \boldsymbol{\varepsilon}_{k-1} + \theta G_{k-1} \dot{\mathbf{y}}_k - l_1 \boldsymbol{\varepsilon}_k - l_2 \sum_{i=0}^k \boldsymbol{\varepsilon}_i. \quad (4.20)$$

Further, expanding G_k through the Taylor expansion [31] leads to

$$G_k = G_{k-1} + \theta \dot{G}_{k-1} + \mathcal{O}(\theta^2) \quad (4.21)$$

where $\mathcal{O}(\theta^2)$ is the matrix with entries of order $\mathcal{O}(\theta^2)$. Substituting (4.21) into (4.20), we have

$$\begin{aligned} \boldsymbol{\varepsilon}_k + \theta (G_{k-1} + \theta \dot{G}_{k-1} + \mathcal{O}(\theta^2)) \dot{\mathbf{y}}_k + \mathcal{O}(\theta^2) = \\ \boldsymbol{\varepsilon}_{k-1} + \theta G_{k-1} \dot{\mathbf{y}}_k - l_1 \boldsymbol{\varepsilon}_k - l_2 \sum_{i=0}^k \boldsymbol{\varepsilon}_i, \end{aligned} \quad (4.22)$$

which yields

$$(1 + l_1) \boldsymbol{\varepsilon}_k - \boldsymbol{\varepsilon}_{k-1} + l_2 \sum_{i=0}^k \boldsymbol{\varepsilon}_i + \mathcal{O}(\theta^2) = \mathbf{0}. \quad (4.23)$$

Thus, the proof is completed.

4.3.2 Convergence of NSDTND Model

Considering the convergence of the designed model (4.10) in the noise-free environment, the following theorem is provided.

Theorem 4.2 *The steady-state error $\lim_{k \rightarrow \infty} \|\boldsymbol{\varepsilon}_k\|_2$ of NSDTND model (4.9) for solving TVNOEIC problem (4.1) converges to $\mathcal{O}(\theta^2)$, with $\|\cdot\|_2$ denoting the 2-norm of a vector.*

Proof From Theorem 4.1, we easily derive

$$(1 + l_1)\mathbf{\varepsilon}_{k+1} - \mathbf{\varepsilon}_k + l_2 \sum_{i=0}^{k+1} \mathbf{\varepsilon}_i + \mathbf{O}(\theta^2) = 0. \quad (4.24)$$

Subtracting the j th subsystem of (4.23) with the j th subsystem of (4.24), it is obtained that

$$(1 + l_1 + l_2)\varepsilon_{k+1}^j = (2 + l_1)\varepsilon_k^j - \varepsilon_{k-1}^j + \mathbf{O}(\theta^2), \quad (4.25)$$

where ε_k^j represents the j th element of $\mathbf{\varepsilon}_k$ with $j = 1, 2, \dots, m + n + p$. Then, Eq. (4.25) can be further written as

$$\phi_{k+1}^j = R\phi_k^j + \mathbf{O}(\theta^2), \quad (4.26)$$

with

$$\phi_{k+1}^j = \begin{bmatrix} \varepsilon_{k+1}^j \\ \varepsilon_k^j \end{bmatrix}, R = \begin{bmatrix} \frac{2+l_1}{1+l_1+l_2} & \frac{-1}{1+l_1+l_2} \\ 1 & 0 \end{bmatrix}.$$

From the above equation, we get that

$$\|\phi_{k+1}^j\|_2 \leq \|R\phi_k^j\|_2 + \|\mathbf{O}(\theta^2)\|_2. \quad (4.27)$$

Considering the nature of $\|\mathbf{O}(\theta^2)\|_2$, i.e., $\|\mathbf{O}(\theta^2)\|_2 = \mathbf{O}(\theta^2)$, we have

$$\begin{aligned} \|\phi_{k+1}^j\|_2 &\leq \|R\phi_k^j\|_2 + \mathbf{O}(\theta^2) \\ &\leq \|R\|_2 \|\phi_k^j\|_2 + \mathbf{O}(\theta^2) \\ &\leq \|R\|_2 \|R\phi_{k-1}^j\|_2 + \|R\|_2 \mathbf{O}(\theta^2) + \mathbf{O}(\theta^2), \end{aligned} \quad (4.28)$$

which can be further derived as

$$\begin{aligned} \|\phi_{k+1}^j\|_2 &\leq \|R\|_2 \|R\phi_{k-1}^j\|_2 + \mathbf{O}(\theta^2) \\ &\leq \|R\|_2^2 \|\phi_{k-1}^j\|_2 + \mathbf{O}(\theta^2) \\ &\vdots \\ &\leq \|R\|_2^k \|\phi_1^j\|_2 + \mathbf{O}(\theta^2). \end{aligned} \quad (4.29)$$

The eigenvalues of matrix R can be obtained, i.e.,

$$p_1 = \frac{l_1 - \sqrt{l_1^2 - 4l_2} + 2}{2l_1 + 2l_2 + 2}, p_2 = \frac{l_1 + \sqrt{l_1^2 - 4l_2} + 2}{2l_1 + 2l_2 + 2}.$$

It is easy to see that the real part of eigenvalues is less than 1. Thus, we can deduce that $\lim_{k \rightarrow \infty} \|R\|_2^k = 0$. Given (4.29), we have

$$\lim_{k \rightarrow \infty} \|\phi_{k+1}^j\|_2 \leq \lim_{k \rightarrow \infty} \|R\|_2^k \|\phi_1^j\|_2 + O(\theta^2) = O(\theta^2). \quad (4.30)$$

Thus, according to the definition of ϕ_{k+1}^j , the steady-state residual error $\lim_{k \rightarrow \infty} \|\epsilon_k\|_2$ of NSDTND model (4.9) for solving the TVNOEIC problem (4.1) is a second-order infinitesimal term $O(\theta^2)$, which means that the neural state of (4.9) globally converges to the theoretical solution of TVNOEIC problem (4.1) with the error being $O(\theta^2)$. The proof is thus completed.

4.3.3 Convergence Under Noisy Condition

The following theoretical analysis is offered to demonstrate the convergence performance of the constructed NSDTND model (4.10) under the constant noise condition.

Theorem 4.3 *In the constant noise $\xi(t) = \eta$ polluted environment, the steady-state error $\lim_{k \rightarrow \infty} \|\epsilon_k\|_2$ of NSDTND model (4.10) for solving the TVNOEIC problem (4.1) converges to $O(\theta^2)$.*

Proof There are two kinds of factors that affect the convergence performance of NSDTND model (4.10), i.e., the constant noise $\xi(t) = \eta$ and the second-order $O(\theta^2)$ error.

It is derived from (4.23) that

$$\epsilon_k = \frac{\epsilon_{k-1} - l_2 \sum_{i=0}^k \epsilon_i}{1 + l_1} + O(\theta^2). \quad (4.31)$$

Considering the linear property of NSDTND model (4.10), these two mentioned factors are analyzed separately. Note that the residual error of (4.31) has been calculated in Theorem 4.2, i.e., $O(\theta^2)$. Therefore, the remaining work is to find the residual error of the following system:

$$(1 + l_1)\epsilon_k = \epsilon_{k-1} - l_2 \sum_{i=0}^k \epsilon_i + \eta, \quad (4.32)$$

which can be expressed as

$$(1 + l_1 + l_2)\epsilon_{k+1} = \epsilon_k - l_2 \sum_{i=0}^k \epsilon_i + \eta. \quad (4.33)$$

With this, it is easy to get the j th subsystem of Eq. (4.33), that is,

$$(1 + l_1 + l_2)\varepsilon_{k+1}^j = \varepsilon_k^j - l_2 \sum_{i=0}^k \varepsilon_i^j + \eta^j. \quad (4.34)$$

Applying Z-transform to Eq. (4.34) yields

$$(1 + l_1 + l_2)(z\varepsilon^j(z) - z\varepsilon^j(0)) = \varepsilon^j(z) + \frac{z\eta^j}{z-1} - l_2 \frac{z\varepsilon^j(z)}{z-1}, \quad (4.35)$$

with $\varepsilon^j(0)$ being the initial value of ε^j . Further, Eq. (4.35) is written as

$$\varepsilon^j(z) = \frac{(1 + l_1 + l_2)z(z-1)\varepsilon^j(0) + z\eta^j}{(1 + l_1 + l_2)z(z-1) - (z-1) + zl_2}. \quad (4.36)$$

The poles of the above equation are easily obtained as

$$z_1 = \frac{l_1 + \sqrt{l_1^2 - 4l_2} + 2}{2l_1 + 2l_2 + 2}, z_2 = \frac{l_1 - \sqrt{l_1^2 - 4l_2} + 2}{2l_1 + 2l_2 + 2}.$$

It is clear that the poles are within a unit circle, which means (4.34) is stable. Applying the final value theorem, we further have

$$\begin{aligned} \lim_{k \rightarrow \infty} \varepsilon_k^j &= \lim_{z \rightarrow 1} (z-1)\varepsilon^j(z) \\ &= \lim_{z \rightarrow 1} \frac{(z-1)((1 + l_1 + l_2)z(z-1)\varepsilon^j(0) + z\eta^j)}{(1 + l_1 + l_2)z(z-1) + zl_2 - (z-1)} \\ &= 0, \end{aligned} \quad (4.37)$$

from which one can deduce that the steady-state error $\lim_{k \rightarrow \infty} \|\varepsilon_k\|_2$ of system (4.32) is zero. As stated before, the steady-state error of system (4.31) is $O(\theta^2)$. According to the superposition theorem, the steady-state error of NSDTND model (4.10) with constant noise $\xi(t) = \eta$ is $O(\theta^2)$. The proof is thus completed.

Moreover, regarding the linear noise $\xi(t_k) = \lambda t + \eta$, it is easy to be calculated that the corresponding residual error is $\|\lambda\|_2 + O(\theta^2)$. The proof is thus complete.

Further, the following theorem discusses the convergence performance of NSDTND model (4.10) when faced with the bounded random noise.

Theorem 4.4 *Under the random noise $\xi(t) = \mu$ polluted environment, the steady-state error $\lim_{k \rightarrow \infty} \|\varepsilon_k\|_2$ of NSDTND model (4.10) for solving the TVNOEIC problem is convergent, and its value holds an upper limit $2m \sup_{1 \leq i \leq k, 1 \leq j \leq m} |\mu_i^j| / (1 - \|R\|_2) + O(\theta^2)$.*

Proof As described in Theorem 4.3, the discussion can be divided into two parts: term $O(\theta^2)$ and random noise term $\xi(t) = \mu$. Similar to Theorem 4.2, we get the equation shown as

$$(1 + l_1 + l_2)\varepsilon_{k+1}^j = (2 + l_1)\varepsilon_k^j - \varepsilon_{k-1}^j + \mu_k^j - \mu_{k-1}^j. \quad (4.38)$$

Rewriting Eq. (4.38) yields $\phi_{k+1}^j = R\phi_k^j + \gamma_k^j$ where $\gamma_k^j = [\mu_k^j - \mu_{k-1}^j, 0]^T$, and the definitions of ϕ_{k+1}^j and R are the same as those in Theorem 4.2. Then, it is derived that

$$\|\phi_{k+1}^j\|_2 \leq \|R\phi_k^j\|_2 + \|\gamma_k^j\|_2, \quad (4.39)$$

that is,

$$\|\phi_{k+1}^j\|_2 \leq \|R\|_2 \|\phi_k^j\|_2 + \|\gamma_k^j\|_2, \quad (4.40)$$

from which the following result is obtained:

$$\begin{aligned} \|\phi_{k+1}^j\|_2 &\leq \|R\|_2 \|\phi_{k-1}^j\|_2 + \|R\|_2 \|\gamma_{k-1}^j\|_2 + \|\gamma_k^j\|_2 \\ &\leq \|R\|_2^2 \|\phi_{k-1}^j\|_2 + \|R\|_2 \|\gamma_{k-1}^j\|_2 + \|\gamma_k^j\|_2 \\ &\quad \vdots \\ &\leq \|R\|_2^k \|\phi_1^j\|_2 + \|R\|_2^{k-1} \|\gamma_1^j\|_2 + \cdots + \|\gamma_k^j\|_2 \\ &< \|R\|_2^k \|\phi_1^j\|_2 + \max_{1 \leq i \leq k} \|\gamma_i^j\|_2 / (1 - \|R\|_2) \\ &< \|R\|_2^k \|\phi_1^j\|_2 + 2 \max_{1 \leq i \leq k} |\mu_i^j| / (1 - \|R\|_2). \end{aligned} \quad (4.41)$$

From the proof Theorem 4.2, we get $\lim_{k \rightarrow \infty} \|R\|_2^k = 0$, and substituting it into (4.41) yields

$$\lim_{k \rightarrow \infty} \|\phi_{k+1}^j\|_2 < 2 \max_{1 \leq i \leq k} |\mu_i^j| / (1 - \|R\|_2). \quad (4.42)$$

Finally, considering the influence of truncation error $O(\theta^2)$, we have

$$\lim_{k \rightarrow \infty} \|\varepsilon_k\|_2 < 2m \sup_{1 \leq i \leq k, 1 \leq j \leq m} |\mu_i^j| / (1 - \|R\|_2) + O(\theta^2). \quad (4.43)$$

Therefore, the proof on the convergence performance of NSDTND model (4.10) under the random noise polluted condition is thus completed.

4.4 Illustrative Examples

The previous section investigates the convergence performance of NSDTND model (4.10) in the presence of different noises through theoretical analyses. In this section, the convergence performance of NSDTND model (4.10) is illustrated by numerical experiments in various noisy conditions, and its superiority is validated by two comparative numerical experiments. In addition, the application of NSDTND model (4.10) to the WTA problem is provided as well.

4.4.1 TVNOEIC Problem-Solving

The effectiveness of NSDTND model (4.10) is verified through numerical simulation. Firstly, consider the following example:

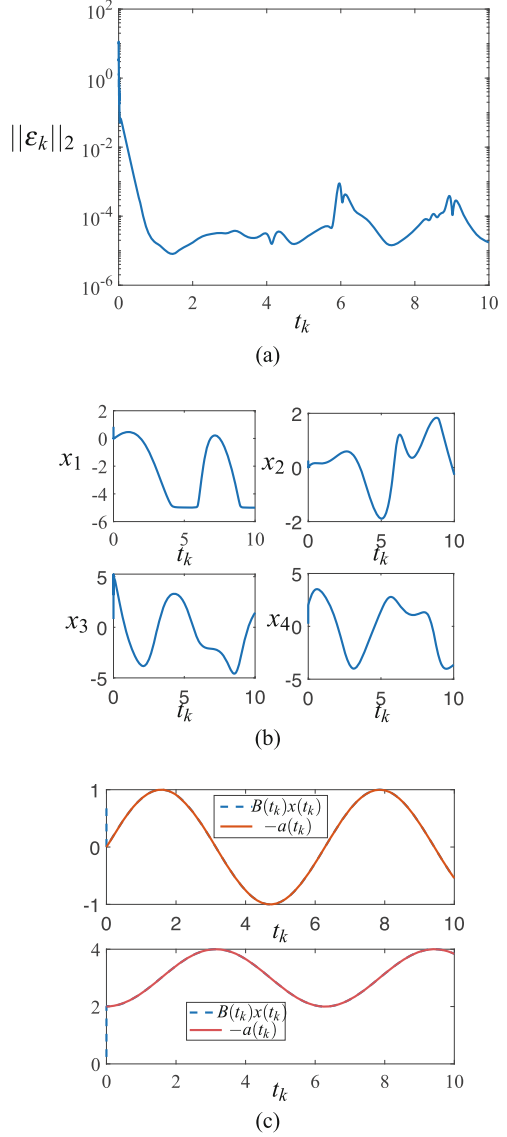
$$\begin{aligned}
 & \min (x_1(t_k) + t_k)^2 + (x_2(t_k) + t_k)^2 + (x_3(t_k) - 5 \exp(-t_k))^2 \\
 & \quad + (x_4(t_k) + \exp(-t_k) - 2)^2 + x_3(t_k) \sin(t_k) \\
 & \quad - (x_1(t_k) + \ln(0.1t_k + 1))(0.1x_2(t_k) + \sin(t_k)) \\
 & \quad + 0.1(t_k - 1)(x_3(t_k) + 5)(x_4(t_k) + 5), \\
 & \text{s. t. } x_1(t_k)(0.5 + \sin(t_k)) + x_2(t_k)(2 - \cos(t_k)) - \sin(t_k) = 0, \\
 & \quad - x_3(t_k) \sin(t_k) + x_4(t_k) \cos(t_k) + \cos(t_k) - 3 = 0, \\
 & \quad - 5 \leq x_1(t_k), x_2(t_k), x_3(t_k), x_4(t_k) \leq 5.
 \end{aligned} \tag{4.44}$$

This problem can be written into a standard formulation as (4.1), of which the coefficients are shown as follows:

$$\begin{aligned}
 B(t_k) &= \begin{bmatrix} \sin(t_k) + 0.5 - \cos(t_k) + 2 & 0 & 0 \\ 0 & 0 & -\sin(t_k) \cos(t_k) \end{bmatrix}, \\
 D(t_k) &= \begin{bmatrix} \mathbf{I} \\ -\mathbf{I} \end{bmatrix}, \mathbf{a}(t_k) = \begin{bmatrix} -\sin(t_k) \\ \cos(t_k) - 3 \end{bmatrix}, \text{ and } \mathbf{c}(t_k) = [-5]^{8 \times 1}, \\
 \mathbf{x}(t_k) &= [x_1(t_k), x_2(t_k), x_3(t_k), x_4(t_k)]^T.
 \end{aligned}$$

All the simulation experiments in this chapter are conducted on MATLAB platform. In this simulation, the starting states are randomly initiated and limited to the range $[0, 1]$. The solving time is set to $[k\theta, (k + 1)\theta] \in [0, 10]$ with $k = 0, 1, 2, \dots$. It can be easily obtained that the problem to be solved (i.e., (4.44)) is convex in the given time range. Besides, the parameters l_1 and l_2 are set as 0.1

Fig. 4.1 Performance of NSDTND model (4.10) for solving TVNOEIC problem (4.44) in the noise-free environment. (a) Residual errors, (b) state variables, and (c) equality constraints



and 0.001, respectively. The residual error, i.e., $\|\epsilon_k\|_2$, is chosen as the evaluation index on the effectiveness of the newly presented model.

Firstly, Fig. 4.1 provides the profiles of residual errors, state variables $(x_1, \dots, x_4)$, and equality constraints of NSDTND model (4.10) for solving problem (4.44) in the absence of noise with $\theta = 0.001$ s. As seen from Fig. 4.1a, the residual error rapidly drops to the steady state, and the maximal value is of order 10^{-4} , demonstrating the effectiveness of NSDTND model (4.10) in solving

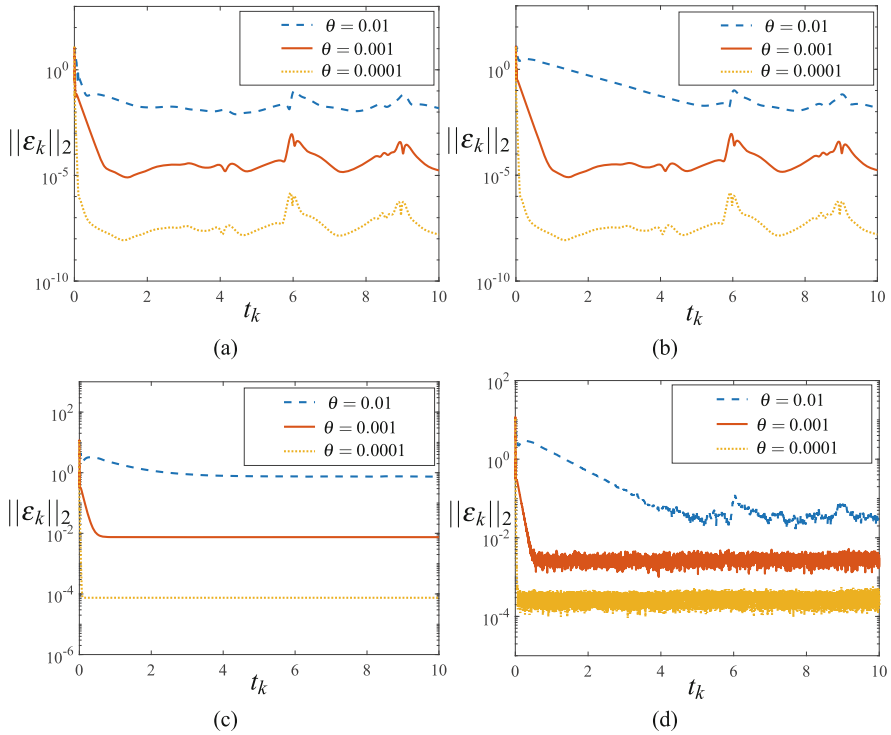


Fig. 4.2 Residual errors of NSDTND model (4.10) for solving TVNOEIC problem (4.44) with different sampling gaps. (a) In the noise-free environment, (b) in the constant noise $\xi_i(t_k) = 10$ polluted environment, (c) in the linear noise $\xi_i(t_k) = 2t_k + 10$ polluted environment, and (d) in the random noise $\xi(t_k) \in [9, 11]$ polluted environment

TVNOEIC problems. As it appears in Fig. 4.1b, these state variables are all strictly within the inequality constraint. In Fig. 4.1c, the transient behaviors of vectors $B(t)x(t)$ and $-a(t)$ begin to overlap in a very short time, and they stay that way until the end, which means that the equality constraint is satisfied.

To explore the impact of different kinds of noises and sampling interval θ on the given model, the residual errors $\|\epsilon(t_k)\|_2$ generated by NSDTND model (4.10) with different noises and different values of θ are shown in Fig. 4.2. The detailed analyses are as follows:

- With noise-free: The corresponding residual errors generated by NSDTND model (4.9) in the absence of noises with sampling gaps $\theta = 0.01$ s, $\theta = 0.001$ s, and $\theta = 0.0001$ s are depicted in Fig. 4.2a, whose maximal steady-state values hold at the orders of 10^{-1} , 10^{-4} , and 10^{-6} , respectively. This result confirms the theoretical result of Theorem 4.2.
- With constant noise: As visualized in Fig. 4.2b, the residual errors of NSDTND model (4.10) for solving TVNOEIC problem (4.44) under the pollution of

constant noise $\xi_i(t_k) = 10$ are convergent. Specifically, the maximal steady-state residual errors are at the orders of 10^{-1} , 10^{-4} , and 10^{-6} with sampling gaps $\theta = 0.01$ s, $\theta = 0.001$ s, and $\theta = 0.0001$ s, respectively, verifying the noise tolerance of NSDTND model (4.10) as illustrated in Theorem 4.3.

- With linear noise: The convergence trends of NSDTND model (4.10) perturbed by linear noise $\xi_i(t_k) = 2t_k + 10$ with different sampling gaps are displayed in Fig. 4.2c which reflects that the residual errors can still converge to the steady states, satisfying the derivation of Theorem 4.3.
- With random noise: As seen from Fig. 4.2d, when faced with random noise $\xi(t_k) \in [9, 11]$, the maximal values of steady-state residual errors synthesized by NSDTND model (4.10) with $\theta = 0.01$ s, $\theta = 0.001$ s, and $\theta = 0.0001$ s are about 10^{-1} , 10^{-3} , and 10^{-4} , respectively. Thus, the correctness of Theorem 4.4 is verified.

4.4.2 Comparisons

Figure 4.3 investigates the superiority of the designed NSDTND model (4.10) compared with the NI model (4.11). Also, the parameters are set as $l_1 = 0.1$, $l_2 = 0.001$, noises are assumed to be zero, and the starting states are randomly initiated in the range $[0, 1]$. As plotted in Fig. 4.3a, the residual error generated by the NSDTND model (4.10) (i.e., of order 10^{-4}) is smaller than that of the NI model (4.11) (i.e., of order 10^{-2}). From Fig. 4.3b, it is observed that the residual errors vary in an $O(\theta)$ manner as θ changes, not as fast as NSDTND model (4.10) does. Therefore, the results suggest that the constructed NSDTND model (4.10) is superior to NI (4.11).

Figure 4.4 exhibits the state variables synthesized by the MNI (4.12) model for solving problem (4.44). As shown, the state variables are neither constrained by equality nor constrained by inequality, which further implies the superiority of the presented model.

4.4.3 WTA Application

This subsection shows the application of NSDTND model (4.10) to WTA problem.

WTA refers to the phenomenon of agents in a group competing for activation, with only those who possess the largest values remaining active and the others inactive, which widely exists in the society and is usually described as the following formulation:

$$z_i(t) = g(q_i(t)) = \begin{cases} 1, & q_i(t) \in \{e \text{ largest elements of } \mathbf{q}(t)\}, \\ 0, & \text{otherwise,} \end{cases} \quad (4.45)$$

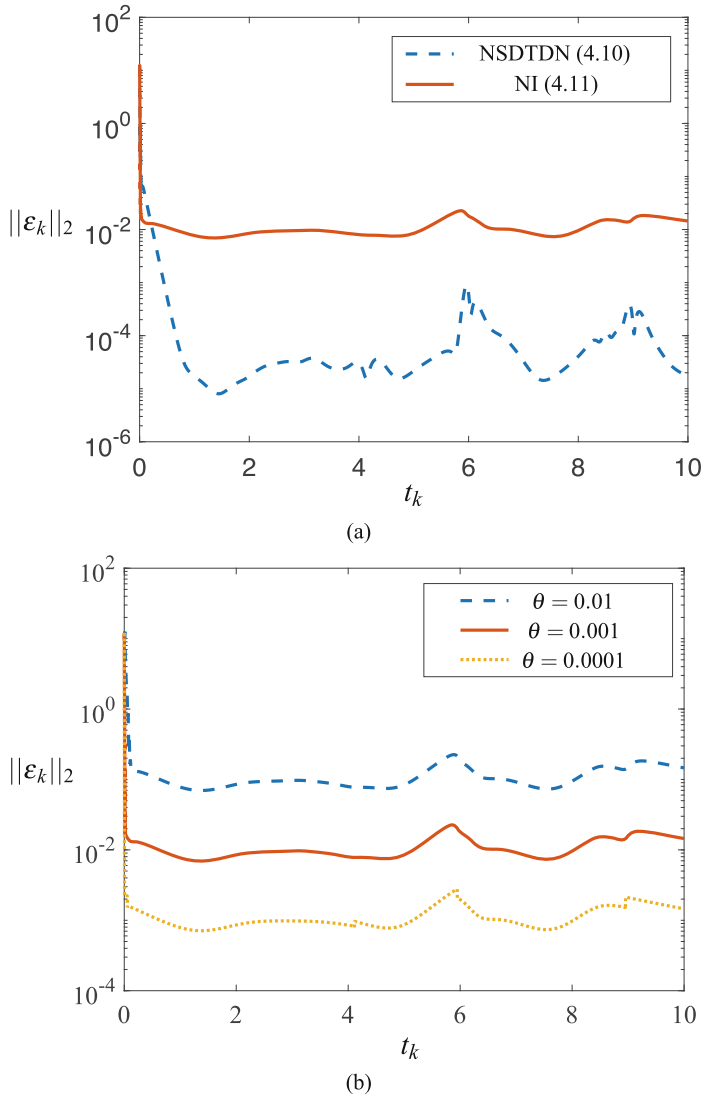


Fig. 4.3 Residual errors of NSDTDND model (4.10) and NI model (4.11) for solving TVNOEIC problem (4.44). (a) By NSTDND model (4.10) and NI model (4.11) with $\theta = 0.001$ and (b) by NI model (4.11) with different sampling gaps

where $q_i(t)$ implies the i th element of the input vector $\mathbf{q}(t) \in \mathbb{R}^m$ with m denoting the number of agents in the group; e represents the number of agents that need to be activated; $z_i(t)$ indicates the i th element of output vector $\mathbf{z}(t)$. Interpreted from (4.45), WTA means finding e maximal elements (i.e., e maximal agents) from

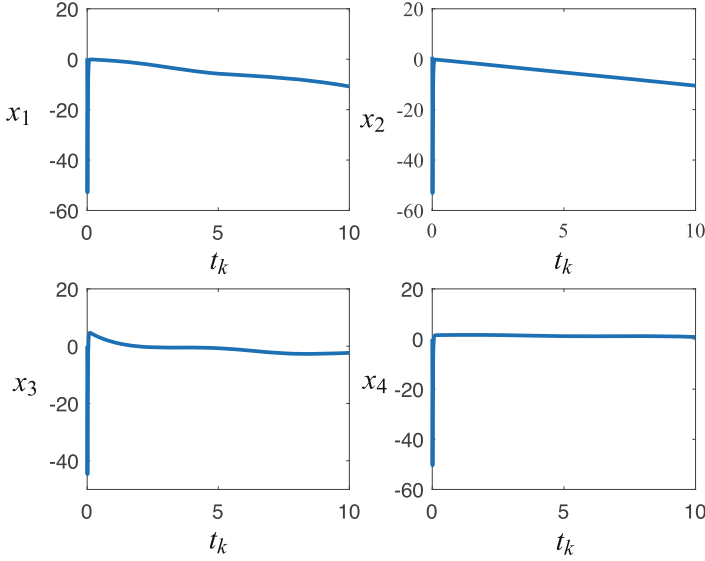


Fig. 4.4 State variables synthesized by MNI model (4.12) for solving (4.44)

vector $\mathbf{q}(t)$ (containing m agents) and setting their values as 1 (i.e., active) and the remaining elements as 0 (i.e., remain inactivated).

Following the formulation in [32, 33], the WTA problem is transformed into a QP problem, which is a special kind of TVNOEIC problem, i.e.,

$$\begin{aligned}
 \min \quad & \varsigma \mathbf{z}^T(t) \mathbf{z}(t) - \mathbf{q}^T(t) \mathbf{z}(t), \\
 \text{s. t.} \quad & \mathbf{a}(t) + B(t) \mathbf{z}(t) = 0, \\
 & \mathbf{c}(t) + D(t) \mathbf{z}(t) \leq 0,
 \end{aligned} \tag{4.46}$$

where ς is a constant coefficient in the range $(0, \tilde{\mathbf{q}}_j)$ with $\tilde{\mathbf{q}}_j$ being the difference between the e th and $(e + 1)$ th largest elements. In this formula,

$$n = 1, p = 2m, \mathbf{a}(t) = -e, B(t) = [1]^{1 \times m},$$

$$D(t) = \begin{bmatrix} \mathbf{I}^m \\ -\mathbf{I}^m \end{bmatrix}, \mathbf{c}(t) = \begin{bmatrix} -\mathbf{1}^m \\ \mathbf{0}^m \end{bmatrix}$$

where $\mathbf{I}^m$ implies the m -dimensional identity matrix; $-\mathbf{1}^m = [-1]^{m \times 1}$; $\mathbf{0}^m = [0]^{m \times 1}$. To investigate and validate the feasibility and the robustness of the given NSDTND (4.10), the time-variant input vector $q_i(t_k) = -\sin(0.25\pi(t_k + 0.8(2*i - 1)))$, ($i = 1, 2, 3, 4$) is considered with the parameters assigned as $e = 2$, $m = 4$, and the initial state values set as a zero vector. The whole simulations are conducted

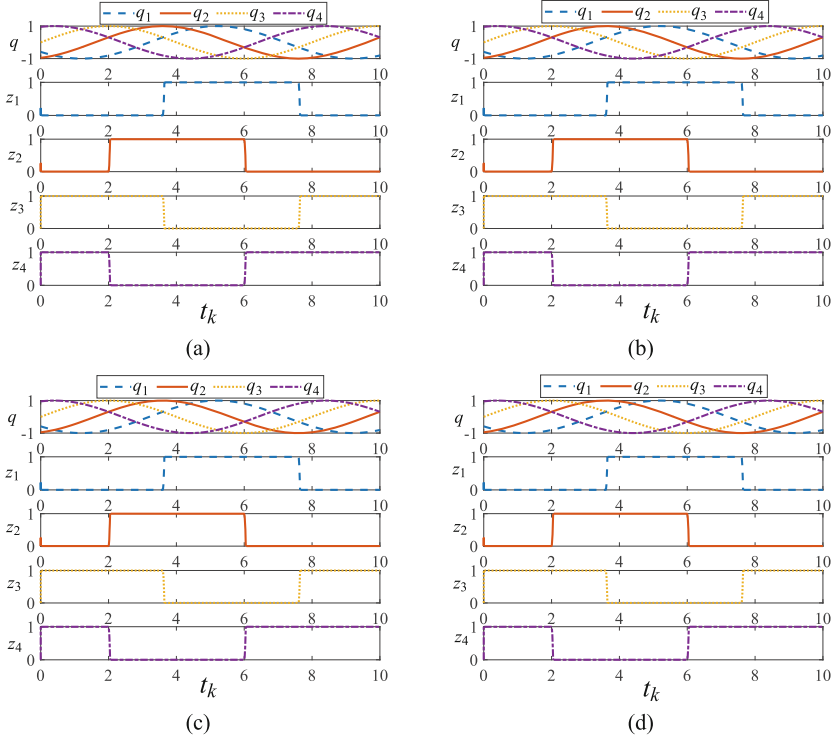


Fig. 4.5 Inputs and outputs of NSDTND model (4.10) for solving a specific WTA problem (4.46). (a) In the noise-free environment, (b) in the constant noise $\xi_i(t_k) = 10$ polluted environment, (c) in the linear noise $\xi_i(t_k) = 2t_k + 10$ polluted environment, and (d) in the random noise $\xi(t_k) \in [9, 11]$ polluted environment

in 10 s and in four scenarios which are distinguished by different kinds of noises. In addition, the real-time behaviors of the four agents and outputs synthesized by NSDTND model (4.10) under four conditions are depicted in Fig. 4.5.

For the noise-free case, as shown in Fig. 4.5a, the two largest inputs correspond to steady-state outputs 1, and the remaining two correspond to outputs 0. In other words, these agents with two largest values are activated, which verifies the effectiveness of NSDTND model (4.10). Furthermore, Fig. 4.5b, c, and d display the transient behaviors of the four agents and outputs produced by NSDTND model (4.10) perturbed by constant noise $\xi_i(t_k) = 10$, linear noise $\xi_i(t_k) = 2t_k + 10$, and random noise $\xi(t_k) \in [9, 11]$, respectively. It is evident that the simulation results are all almost the same as those in Fig. 4.5a, demonstrating the robustness of the designed NSDTND model (4.10).

Overall, the constructed NSDTND model (4.10) possesses the advantages of satisfactory computational accuracy, excellent noise-suppressing ability, and strong practicability as exhibited in the simulation results, which are superior to existing methods.

4.5 Conclusions

In this chapter, an NSDTND model (4.10) has been presented and investigated to handle TVNOEIC problem (4.1) by exploiting the Euler discretization formula. It has been concluded from the theoretical analyses that NSDTND model has not only high accuracy but also strong robustness in the presence of the constant noises, linear noises, and random noises. In addition, illustrative numerical experiments have been conducted in noisy and noise-free environments with different sampling gaps, which verifies the effectiveness of the given NSDTND model. Further, the superiority of the designed model is substantiated by comparative simulation results.

References

1. Xie Z, Jin L, Du X, Xiao X, Li H, Li S (2019) On generalized RMP scheme for redundant robot manipulators aided with dynamic neural networks and nonconvex bound constraints. *IEEE Trans Ind Inform* 15(9):5172–5181
2. Zhang Z, Kong L, Zheng L, Zhang P, Qu X, Liao B, Yu Z (2020) Robustness analysis of a power-type varying-parameter recurrent neural network for solving time-varying QM and QP problems and applications. *IEEE Trans Syst Man Cybern* 50(12):5106–5118
3. Hu X, Zhang B (2009) A new recurrent neural network for solving convex quadratic programming problems with an application to the k-winners-take-all problem. *IEEE Trans Neural Netw Learn Syst* 20(4):654–664
4. Li S, Zhou M, Luo X, You Z (2017) Distributed winner-take-all in dynamic networks. *IEEE Trans Autom Control* 62(2):577–589
5. Zhang Y, Li S, Liao L (2021) Consensus of high-order discrete-time multiagent systems with switching topology. *IEEE Trans Syst Man Cybern* 51(2):721–730
6. Yesikov O, Yesikov D, Rumiantsev V, Ivutin A (2019) Procedure of selecting stochastic search multi-agent algorithm parameters by the example of solving discrete optimization problems. In: 2019 8th Mediterranean conference on embedded computing (MECO), pp 1–4
7. Kement CE, Gultekin H, Tavli B, Girici T, Uludag S (2017) Comparative analysis of load shaping based privacy preservation strategies in smart grid. *IEEE Trans Ind Inform* 13(6):3226–3235
8. Sun T, Sun X-M (2021) An adaptive dynamic programming scheme for nonlinear optimal control with unknown dynamics and its application to turbofan engines. *IEEE Trans Ind Inform* 17(1):367–376
9. Ran X, Liu K (2019) Robust scatter index method for the appliances scheduling of home energy local network with user behavior uncertainty. *IEEE Trans Ind Inform* 15(7):4129–4139
10. Stanimirovic P, Ivanov B, Ma H, Masic D (2020) A survey of gradient methods for solving nonlinear optimization. *Electron Res Arch* 28(4):1573–1624
11. Nazemi A, Nazemi M (2014) A gradient-based neural network method for solving strictly convex quadratic programming problems. *Cogn Comput* 6(3):484–495
12. Chai R, Tsourdos A, Savvaris A, Chai S, Xia Y (2019) Two-stage trajectory optimization for autonomous ground vehicles parking maneuver. *IEEE Trans Ind Inform* 15(7):3899–3909
13. Li D, Fukushima M (2001) A modified BFGS method and its global convergence in nonconvex minimization. *J Comput Appl Math* 129:15–35
14. Huang H, Fu D, Wang G, Jin L, Liao S, Wang H (2021) Modified Newton integration algorithm with noise suppression for online dynamic nonlinear optimization. *Numer Algorithms* 87:575–599

15. Yang C, Jiang Y, He W, Na J, Xu B (2018) Adaptive parameter estimation and control design for robot manipulators with finite-time convergence. *IEEE Trans Ind Electron* 65(99):8112–8123
16. Li J, Mao M, Uhlig F, Zhang Y (2017) Z-type neural-dynamics for time-varying nonlinear optimization under a linear equality constraint with robot application. *J Comput Appl Math* 327:155–166
17. Liu R, Yang B, Hauptmann AG (2020) Simultaneous bearing fault recognition and remaining useful life prediction using joint-loss convolutional neural network. *IEEE Trans Ind Inform* 16(1):87–96
18. Cao J, Mao X, Luo Q (2013) Neurodynamic system theory and applications. *Abstract Appl Anal* 1:551–552
19. Chin CS, Ji X, Woo WL, Kwee TJ, Yang W (2019) Modified multiple generalized regression neural network models using fuzzy C-means with principal component analysis for noise prediction of offshore platform. *Neural Comput Appl* 31(4):1127–1142
20. Khan AH, Li S, Luo X (2020) Obstacle avoidance and tracking control of redundant robotic manipulator: an RNN-based metaheuristic approach. *IEEE Trans Ind Inform* 16(7):4670–4680
21. Li Y, Li S, Hannaford B (2019) A model-based recurrent neural network with randomness for efficient control with applications. *IEEE Trans Ind Inform* 15(4):2054–2063
22. Li W, Su Z, Tan Z (2019) A variable-gain finite-time convergent recurrent neural network for time-variant quadratic programming with unknown noises endured. *IEEE Trans Ind Inform* 15(9):5330–5340
23. Xiao L, Tao J, Dai J, Wang Y, Jia L, He Y (2021) A parameter-changing and complex-valued zeroing neural-network for finding solution of time-varying complex linear matrix equations in finite-time. *IEEE Trans Ind Inform* 17(10):6634–6643
24. Jin L, Liu L, Wang, X, Shang, M, Wang, FY (2024) Physical-informed neural network for MPC-based trajectory tracking of vehicles with noise considered. *IEEE Trans Intell Veh*. In Press. <https://doi.org/10.1109/TIV.2024.3358229>
25. Yan J, Jin L, Luo X, Li S (2023) Modified RNN for solving comprehensive Sylvester equation with TDOA application. *IEEE Trans Neural Netw Learn Syst*. In Press. <https://doi.org/10.1109/TNNLS.2023.3263565>
26. Li W, Ma X, Luo J, Jin L (2021) A strictly predefined-time convergent neural solution to equality- and inequality-constrained time-variant quadratic programming. *IEEE Trans Syst Man Cybern Syst* 51(7):4028–4039
27. Zhang X, Chen L, Li S, Stanimirovic P, Zhang J, Jin L (2021) Design and analysis of recurrent neural network models with non-linear activation functions for solving time-varying quadratic programming problems. *CAAI Trans Intell Technol* 6(4):394–404
28. Zhang Z, Chen S, Liang J (2021) Discrete-time circadian rhythms neural network for perturbed redundant robot manipulators tracking problem with periodic noises. *IEEE Trans Ind Inform* 18(1):242–251
29. Guo D, Lin X, Sun S, Huang Z (2018) Design and analysis of two discrete-time ZD algorithms for time-varying nonlinear minimization. *Numer Algorithms* 77(1):23–36
30. Das S, Halder K, Gupta A (2020) Delay handling method in dominant pole placement based PID controller design. *IEEE Trans Ind Inform* 16(2):980–991
31. Mathews JH, Fink KD (2004) Numerical methods using MATLAB. Prentice-Hall, New Jersey
32. Liu S, Wang J (2006) A simplified dual neural network for quadratic programming with its KWTa application. *IEEE Trans Neural Netw Learn Syst* 17(6):1500–1510
33. Liang S, Jin L (2024) Distributed k-winner-take-all network under weight-unbalanced topology. *IEEE Trans Circuits Syst II Express Briefs* 71(2):717–721

Chapter 5

Discrete Computational Neural Dynamics



5.1 Introduction

Sylvester equation is a crucial kind of linear matrix equation in matrix theory, which is broadly encountered in mathematics and control fields, such as image fusions [1], eigen-structure assignment [2], semi-supervised learning [3], and other applications [4–6]. Besides, many problems can be solved by finding the solutions of Sylvester equations in analyses and control theory and thus much effort has been devoted for solving them [7, 8]. The classic solution methods for solving stationary Sylvester equations include the Bartels-Stewart algorithm [7] and the Hessenberg-Schur method [8], which often transform the coefficient matrices to Schur or Hessenberg form and then solve the corresponding linear system of equations directly iteratively. Moreover, the Bartels-Stewart algorithm can be utilized appropriately with its extensions only when the sampling period τ is large enough with a time complexity $O(\tau^3)$ and may not solve time-dependent problems effectively due to the lack of adaption to the time-dependent parameters.

Neural networks are widely reported and employed in scientific and engineering areas [9, 10]. Among them, a great deal of gradient-related neural networks are designed and constructed for solving Sylvester equations, which employ the norm of the error function as the performance indicator and use a negative gradient-descent direction to make the error norm converge to zero with time [11, 12]. In [13], an iterative algorithm is constructed to solve coupled Sylvester transpose matrix equations by extending the conjugate gradient method. However, for solving problems with time-dependent parameters, it is difficult for the gradient method to track theoretical solutions accurately, and the norm-based error function cannot converge to zero in time [14], which highlights the infeasibility of gradient-related neural networks. A recurrent neural dynamics based on the traditional zeroing neural network design formula is developed in [15] for solving time-dependent Sylvester equations, which obtains a better convergence compared with previous approaches but with infinite convergence time to the theoretical solution.

Li et al. design an improved zeroing neural network with a finite-time convergence property [16]. Such a method can successfully solve time-dependent problems [15, 17, 18] and accelerate the convergence speed by utilizing the time derivative of the error function [19–21]. Additionally, the Davidenko method, which extends the limitations on the choice of initial values with an exponential convergence performance, is designed as an improvement of the Newton iteration method and can solve time-dependent nonlinear equation systems effectively [22].

Considering that a dynamical system depicted in continuous form cannot be implemented directly on digital computers and that there always exist measurement noises or external perturbations from environmental disturbance or hardware implementation, having a significant impact on the precision of the solutions, new discrete computational models with high robustness for solving time-dependent Sylvester equations are worth exploiting. In this chapter, we focus on new models by addressing the mentioned demands, with quasi-Newton Broyden-Fletcher-Goldfarb-Shanno (BFGS) leveraged to eliminate the matrix inversion. The remainder of this chapter is organized as follows. Section 5.2 presents the problem formulation. In Sect. 5.2.1, by transforming Sylvester equations into linear equations and defining an error function, the discrete computational neural dynamics (DCND) and DCND-BFGS models are designed and investigated for solving the time-dependent Sylvester equations. Theoretical analyses are further presented in Sect. 5.3 to account for the convergence property of the given models and the superior performance in the presence of perturbations. In addition, Sect. 5.4 provides illustrative examples and two applications on robot manipulators and multiple-input multiple-output (MIMO) systems to substantiate the high efficiency and superiority of the presented models compared with existing models. Section 5.5 concludes this chapter.

5.2 Problem Formulation

Considering the following time-dependent Sylvester equations:

$$M(t)Z(t) - Z(t)N(t) + C(t) = 0, t \in [t_0, t_f] \quad (5.1)$$

where $M(t) \in \mathbb{R}^{n \times n}$, $N(t) \in \mathbb{R}^{p \times p}$, and $C(t) \in \mathbb{R}^{n \times p}$ are the time-dependent coefficient matrices; t_0 and t_f represent the initial time and final time, respectively. In this chapter, we assume that (5.1) always has a unique solution at any time instant t , which is guaranteed with the following condition shown in Lemma 5.1.

Lemma 5.1 *Time-dependent Sylvester equation (5.1) has a unique solution only if $M(t)$ and $N(t)$ have no identical eigenvalues; in this sense, $\sigma(M(t)) \cap \sigma(N(t)) = \emptyset$ ([23], Theorem 2.4.4.1).*

Equation (5.1) can be further written as the following time-dependent linear equation (TDLE):

$$Q(t)\mathbf{y}(t) = \mathbf{a}(t), \quad (5.2)$$

where $Q(t) = M(t) \otimes I_p - I_n \otimes N(t) \in \mathbb{R}^{np \times np}$; $\mathbf{y}(t) = \text{vec}(Z(t)) \in \mathbb{R}^{np}$; $\mathbf{a}(t) = -\text{vec}(C(t)) \in \mathbb{R}^{np}$ with $\otimes$ the Kronecker product and $\text{vec}(\cdot)$ the matrix vectorization operation. In view of the above equivalent transformation, the unique solution of (5.1) can be obtained by solving (5.2) with the theoretical solution generated as

$$\mathbf{y}^*(t) = Q^{-1}(t)\mathbf{a}(t) \in \mathbb{R}^{np},$$

where superscript $^{-1}$ denotes the inversion operation. This theoretical solution $\mathbf{y}^*(t)$ and its corresponding $Z^*(t)$ of time-dependent Sylvester equations (5.1) are used to validate the efficacy of the constructed discrete computational neural dynamics models. The vectors or matrices mentioned above are time-dependent and time instant $t \in [t_0, t_f]$ can be expressed as $t = k\tau$ with τ denoting the sampling interval and $k = 0, 1, 2, \dots$ representing the updating index. It is emphasized that the solution $\mathbf{y}((k+1)\tau)$ should be calculated before the time instant $t = (k+1)\tau$ while the data $Q((k+1)\tau)$, $\mathbf{a}((k+1)\tau)$ are not attainable till the time instant $t = (k+1)\tau$ in the practical computing. In this sense, what we do is to conduct a calculation or forecast of the future solution aided by the current and/or former data.

5.2.1 Redundant Robot Manipulator

In this section, two discrete computational neural dynamics (DCND) models are constructed and investigated for solving time-dependent Sylvester equations (5.1).

5.2.2 Description of the DCND Model

To develop the DCND model for solving time-dependent linear equation (5.2), firstly, a vector-valued error function is defined as follows:

$$\boldsymbol{\epsilon}(t) = Q(t)\mathbf{y}(t) - \mathbf{a}(t) \in \mathbb{R}^{np}. \quad (5.3)$$

Secondly, a design formula of error function $\boldsymbol{\epsilon}(t)$ is described as

$$\dot{\boldsymbol{\epsilon}}(t) = -\phi\boldsymbol{\epsilon}(t) - \psi \int_0^t \boldsymbol{\epsilon}(\delta)d\delta. \quad (5.4)$$

Finally, combining equations (5.3) and (5.4) generates

$$Q(t)\dot{\mathbf{y}}(t) = -\dot{Q}(t)\mathbf{y}(t) + \dot{\mathbf{a}}(t) - \phi(Q(t)\mathbf{y}(t) - \mathbf{a}(t)) - \psi \int_0^t \boldsymbol{\epsilon}(\delta) d\delta. \quad (5.5)$$

Considering the impact of noises on computation process, the above equation can be rearranged as follows:

$$Q(t)\dot{\mathbf{y}}(t) = -\dot{Q}(t)\mathbf{y}(t) + \dot{\mathbf{a}}(t) - \phi(Q(t)\mathbf{y}(t) - \mathbf{a}(t)) - \psi \int_0^t \boldsymbol{\epsilon}(\delta) d\delta + \mathcal{G}(t), \quad (5.6)$$

where $\mathcal{G}(t)$ signifies the measurement noises. Aided with the Euler forward difference formula $\dot{\mathbf{y}}_k = (\mathbf{y}_{k+1} - \mathbf{y}_k)/\tau$, equation (5.5) can be discretized clearly as the following, which is the designed DCND model:

$$\begin{aligned} \mathbf{y}_{k+1} = & \mathbf{y}_k + Q_k^{-1}(-(Q_k - Q_{k-1})\mathbf{y}_k + \mathbf{a}_k - \mathbf{a}_{k-1} - \\ & \phi\tau(Q_k\mathbf{y}_k - \mathbf{a}_k) - \psi\tau \sum_{i=0}^k (Q_i\mathbf{y}_i - \mathbf{a}_i)), \end{aligned} \quad (5.7)$$

where the neurally computed solution $\mathbf{y}_k \in \mathbb{R}^{np}$ corresponds to $Z(t) \in \mathbb{R}^{n \times p}$ of time-dependent Sylvester equations (5.1) at time instant $t = k\tau$, with a randomly generated initial value $\mathbf{y}_0 \in \mathbb{R}^{np}$. Alternatively, let $\vartheta = \phi\tau$ and $\eta = \psi\tau$; then the noise-polluted DCND model obtained by discretization of equation (5.6) can be rephrased as follows:

$$\begin{aligned} \mathbf{y}_{k+1} = & \mathbf{y}_k + Q_k^{-1}(-(Q_k - Q_{k-1})\mathbf{y}_k + \mathbf{a}_k - \mathbf{a}_{k-1} - \\ & \vartheta(Q_k\mathbf{y}_k - \mathbf{a}_k) - \eta \sum_{i=0}^k (Q_i\mathbf{y}_i - \mathbf{a}_i) + \mathcal{G}_k). \end{aligned}$$

5.2.3 Description of the DCND-BFGS Model

The BFGS algorithm is typically introduced into the problems involving matrix inversion due to its superiority in eliminating the matrix inversion operation, which enlightens us that the BFGS algorithm can be adopted to estimate $Q^{-1}(t)$ if the time-dependent coefficient matrix $Q(t)$ is positive definite and symmetric. Besides, $\tilde{E}_k$ is used to denote the estimated time-dependent inverse of matrix $Q(t)$ at time instant $t = k\tau$. According to the BFGS formula [24–26], the corresponding DCND-BFGS model is generated as follows:

$$\begin{aligned} \mathbf{y}_{k+1} = & \mathbf{y}_k + \tilde{E}_k(-(Q_k - Q_{k-1})\mathbf{y}_k + \mathbf{a}_k - \mathbf{a}_{k-1} - \\ & \phi\tau(Q_k\mathbf{y}_k - \mathbf{a}_k) - \psi\tau \sum_{i=0}^k (Q_i\mathbf{y}_i - \mathbf{a}_i)), \end{aligned} \quad (5.8)$$

where $\tilde{E}_k$ is obtained from the previous $\tilde{E}_{k-1}$ through iteration by using the BFGS method with the recursive iterative expression of $\tilde{E}_k$ presented directly as

$$\Delta \tilde{E}_k = \frac{(\mu_k^T \xi_k + \xi_k^T \tilde{E}_k \xi_k)(\mu_k \mu_k^T)}{(\mu_k^T \xi_k)^2} - \frac{\tilde{E}_k \xi_k \mu_k^T + \mu_k \xi_k^T \tilde{E}_k}{\mu_k^T \xi_k}, \quad (5.9)$$

of which superscript T denotes the transpose of matrices or vectors; $\mu_k = \mathbf{y}_{k+1} - \mathbf{y}_k$; $\xi_k = Q_k \mu_k$; the iteration relationship is defined as $\tilde{E}_{k+1} = \tilde{E}_k + \Delta \tilde{E}_k$ ($k = 0, 1, 2, 3, \dots$). To ensure the accuracy and convergence performance of the solution procedure, $\tilde{E}_0$ can be simply set as $Q^{-1}(\mathbf{y}_0, t_0)$.

In what follows, for comparison, the Euler-type discrete-time zeroing neural dynamics (EDZND) model based on the continuous-time zeroing neural dynamics model is provided as follows [27]:

$$\mathbf{y}_{k+1} = \mathbf{y}_k + Q_k^{-1}(-\tau \dot{Q}_k \mathbf{y}_k + \tau \dot{\mathbf{a}}_k - \vartheta(Q_k \mathbf{y}_k - \mathbf{a}_k)). \quad (5.10)$$

Besides, the traditional Newton iteration (NI) method for solving time-dependent Sylvester equations (5.1) is formulated as

$$\mathbf{y}_{k+1} = \mathbf{y}_k - Q_k^{-1}(Q_k \mathbf{y}_k - \mathbf{a}_k) = Q_k^{-1} \mathbf{a}_k. \quad (5.11)$$

The Davidenko method is introduced here to solve time-dependent Sylvester equations (5.1) based on [22], which is obtained as follows:

$$\dot{\mathbf{y}}_k = -Q_k^{-1}(Q_k \mathbf{y}_k - \mathbf{a}_k) = -\mathbf{y}_k + Q_k^{-1} \mathbf{a}_k. \quad (5.12)$$

5.2.4 The DCND Model Simplification

The following theorem is presented to construct a simplified form of DCND model (5.7) for further research.

Theorem 5.1 *DCND model (5.7) can be evolved as follows:*

$$(1 + \vartheta + \eta)\boldsymbol{\epsilon}_k - \boldsymbol{\epsilon}_{k-1} + \eta \sum_{i=0}^{k-1} \boldsymbol{\epsilon}_i + \mathbf{O}(\tau^2) = 0,$$

of which $\mathbf{O}(\tau^2)$ denotes a vector composed of $O(\tau^2)$.

Proof DCND model (5.7) can be rearranged as

$$Q_k(\mathbf{y}_{k+1} - \mathbf{y}_k) = \mathbf{a}_k - \mathbf{a}_{k-1} - (Q_k - Q_{k-1})\mathbf{y}_k - \vartheta \boldsymbol{\epsilon}_k - \eta \sum_{i=0}^k \boldsymbol{\epsilon}_i. \quad (5.13)$$

Simplifying the above equation can obtain

$$Q_k \mathbf{y}_{k+1} = \mathbf{a}_k - \mathbf{a}_{k-1} + Q_{k-1} \mathbf{y}_k - \vartheta \boldsymbol{\epsilon}_k - \eta \sum_{i=0}^k \boldsymbol{\epsilon}_i. \quad (5.14)$$

Applying the Taylor expansion expression to $\mathbf{y}$ yields

$$\mathbf{y}_{k+1} = \mathbf{y}_k + \tau \dot{\mathbf{y}}_k + \mathbf{O}(\tau^2), \quad (5.15)$$

$$\mathbf{y}_{k-1} = \mathbf{y}_k - \tau \dot{\mathbf{y}}_k + \mathbf{O}(\tau^2). \quad (5.16)$$

Equation (5.16) can be rewritten as

$$\mathbf{y}_k = \mathbf{y}_{k-1} + \tau \dot{\mathbf{y}}_k + \mathbf{O}(\tau^2). \quad (5.17)$$

Substituting equations (5.15) and (5.17) into equation (5.14), we get

$$\begin{aligned} Q_k(\mathbf{y}_k + \tau \dot{\mathbf{y}}_k + \mathbf{O}(\tau^2)) &= \mathbf{a}_k - \mathbf{a}_{k-1} + Q_{k-1}(\mathbf{y}_{k-1} + \tau \dot{\mathbf{y}}_k \\ &\quad \mathbf{O}(\tau^2)) - \vartheta \boldsymbol{\epsilon}_k - \eta \sum_{i=0}^k \boldsymbol{\epsilon}_i, \end{aligned}$$

which can be simplified as

$$\begin{aligned} &Q_k \mathbf{y}_k + \tau Q_k \dot{\mathbf{y}}_k + \mathbf{O}(\tau^2) - \mathbf{a}_k \\ &= Q_{k-1} \mathbf{y}_{k-1} + \tau Q_{k-1} \dot{\mathbf{y}}_k - \mathbf{a}_{k-1} - \vartheta \boldsymbol{\epsilon}_k - \eta \sum_{i=0}^k \boldsymbol{\epsilon}_i. \end{aligned}$$

It can be further reformulated as

$$\boldsymbol{\epsilon}_k + \tau Q_k \dot{\mathbf{y}}_k + \mathbf{O}(\tau^2) = \boldsymbol{\epsilon}_{k-1} + \tau Q_{k-1} \dot{\mathbf{y}}_k - \vartheta \boldsymbol{\epsilon}_k - \eta \sum_{i=0}^k \boldsymbol{\epsilon}_i. \quad (5.18)$$

Using Taylor expansion to Q_k generates

$$Q_k = Q_{k-1} + \tau \dot{Q}_{k-1} + \mathcal{J}(\tau^2), \quad (5.19)$$

where $\mathcal{J}(\tau^2)$ represents a matrix composed of $O(\tau^2)$. Simplifying equation (5.18) aided with equation (5.19), we obtain

$$\begin{aligned} & \epsilon_k + \tau(Q_{k-1} + \tau \dot{Q}_{k-1} + \mathcal{J}(\tau^2))\dot{\mathbf{y}}_k + \mathbf{O}(\tau^2) \\ &= \epsilon_{k-1} + \tau Q_{k-1}\dot{\mathbf{y}}_k - \vartheta \epsilon_k - \eta \sum_{i=0}^k \epsilon_i, \end{aligned}$$

which can be rewritten as

$$(1 + \vartheta + \eta)\epsilon_k - \epsilon_{k-1} + \eta \sum_{i=0}^{k-1} \epsilon_i + \mathbf{O}(\tau^2) = 0. \quad (5.20)$$

The proof is thus completed. $\square$

5.3 Control Theoretical Analyses

In this section, we provide control theoretical analyses to demonstrate the global convergence and robustness in the presence of noises of the given DCND model (5.7) and DCND-BFGS model (5.8). We provide the following Theorem 5.2 with reference to the convergence performance of the presented DCND model (5.7).

Theorem 5.2 *The residual error synthesized by DCND model (5.7) for solving time-dependent Sylvester equations (5.1) converges to $O(\tau^2)$ globally. Videlicet $\lim_{k \rightarrow \infty} \|\epsilon_k\|_2 = O(\tau^2)$, where $\|\cdot\|_2$ denotes the two-norm of a vector.*

Proof As a simplified form of DCND model (5.7), equation (5.20) can be rearranged as

$$(1 + \vartheta)\epsilon_{k+1} - \epsilon_k + \eta \sum_{i=0}^{k+1} \epsilon_i + \mathbf{O}(\tau^2) = 0. \quad (5.21)$$

Letting ϵ_k^l signify the l th ($l \in (1, 2, \dots, np)$) element at time instant $k\tau$ of the above equation yields

$$(1 + \vartheta)\epsilon_{k+1}^l - \epsilon_k^l + \eta \sum_{i=0}^{k+1} \epsilon_i^l + O(\tau^2) = 0. \quad (5.22)$$

Next, letting the l th system at time instant $k\tau$ of (5.21) subtract the l th system at time instant $(k+1)\tau$ of (5.21), we have

$$(1 + \vartheta + \eta)\epsilon_{k+1}^t = (2 + \vartheta)\epsilon_k^t - \epsilon_{k-1}^t + O(\tau^2). \quad (5.23)$$

Let

$$\rho_{k+1}^t = [\epsilon_{k+1}^t, \epsilon_k^t]^T. \quad (5.24)$$

Then, equation (5.23) can be overwritten as

$$\rho_{k+1}^t = \mathcal{L} \cdot \rho_k^t + \mathbf{O}(\tau^2), \quad (5.25)$$

where the matrix $\mathcal{L}$ is defined as

$$\mathcal{L} = \begin{bmatrix} (2 + \vartheta)/(1 + \vartheta + \rho) & -1/(1 + \vartheta + \rho) \\ 1 & 0 \end{bmatrix}.$$

The eigenvalues of the matrix $\mathcal{L}$ are considered as γ_1 and γ_2 , which are $\gamma_{1,2} = (\vartheta + 2 \mp \sqrt{\vartheta^2 - 4\vartheta})/(2\vartheta + 2\rho + 2)$. Expanding (5.25) leads to

$$\begin{aligned} \|\rho_{k+1}^t\|_2 &\leq \|\mathcal{L} \cdot \rho_k^t\|_2 + \|\mathbf{O}(g^2)\|_2 \\ &= \|\mathcal{L} \cdot \rho_k^t\|_2 + O(\tau^2) \\ &\leq \|\mathcal{L}\|_2 \cdot \|\rho_k^t\|_2 + O(\tau^2) \\ &= \|\mathcal{L}\|_2 \cdot \|\mathcal{L} \cdot \rho_{k-1}^t + \mathbf{O}(\tau^2)\|_2 + O(\tau^2) \\ &\leq \|\mathcal{L}\|_2^2 \cdot \|\rho_{k-1}^t\|_2 + O(\tau^2) \\ &= \dots \\ &\leq \|\mathcal{L}\|_2^k \cdot \|\rho_1^t\|_2 + O(\tau^2). \end{aligned}$$

Knowing that $|\text{Re}(\gamma_1)| < 1$ and $|\text{Re}(\gamma_2)| < 1$, it follows that $\lim_{k \rightarrow \infty} \|\mathcal{L}\|_2^k = 0$. Therefore,

$$\lim_{k \rightarrow \infty} \|\rho_{k+1}^t\|_2 = \lim_{k \rightarrow \infty} \|\mathcal{L}\|_2^k \cdot \|\rho_1^t\|_2 + O(\tau^2) = O(\tau^2).$$

We can further get $\lim_{k \rightarrow \infty} \|\epsilon_k^t\|_2 = O(\tau^2)$ based on equation (5.24). The proof is thus completed. $\square$

Hereinafter, we offer theorems with different types of noises considered.

Theorem 5.3 *The residual error for solving time-dependent Sylvester equations (5.1), when the constructed DCND model (5.7) is perturbed by linear time-dependent noise $\mathcal{G}(t) = \lambda t + \varpi$, globally converges to $\|\lambda/\psi\|_2 + O(\tau^2)$.*

Proof It can be extended from Theorem 5.1 that DCND model (5.7) is a linear system, and $\mathcal{G}_k = \lambda k \tau + \varpi$ and $O(\tau^2)$ can be analyzed respectively on the basis of the superposition of a linear system. Therefore, the relationship between residual error and the noise component upon equation (5.20) can be described as

$$(1 + \vartheta + \eta)\epsilon_k = \epsilon_{k-1} - \eta \sum_{i=0}^{k-1} \epsilon_i + \lambda k \tau + \varpi,$$

which can be further formulated as

$$(1 + \vartheta + \eta)\epsilon_{k+1} = \epsilon_k - \eta \sum_{i=0}^k \epsilon_i + \lambda(k+1)\tau + \varpi. \quad (5.26)$$

The ι th subsystem of (5.26) is

$$(1 + \vartheta + \eta)\epsilon_{k+1}^{\iota} = \epsilon_k^{\iota} - \eta \sum_{i=0}^k \epsilon_i^{\iota} + \lambda^{\iota}(k+1)\tau + \varpi^{\iota}. \quad (5.27)$$

Manipulating the above equation with Z-transform elicits

$$(1 + \vartheta + \eta)(z\epsilon^{\iota}(z) - z\epsilon^{\iota}(0)) = \epsilon^{\iota}(z) - \eta \cdot \frac{\epsilon^{\iota}(z)}{1 - z^{-1}} + \frac{z\lambda^{\iota}\tau}{(z-1)^2} + \frac{z(\varpi^{\iota} + \lambda^{\iota}\tau)}{z-1}, \quad (5.28)$$

where $\epsilon^{\iota}(0)$ presents the initial value of $\epsilon^{\iota}(t)$. Reformulating (5.28) derives

$$\begin{aligned} \epsilon^{\iota}(z) &= \frac{z(z-1)^2(1 + \vartheta + \eta)\epsilon^{\iota}(0) + z(z-1)(\varpi^{\iota} + \lambda^{\iota}\tau) + z\lambda^{\iota}\tau}{z(z-1)^2(1 + \vartheta + \eta) - (z-1)^2 + \eta z(z-1)}. \end{aligned}$$

The poles of the above equation are $z_1 = 1$, $z_{2,3} = (\vartheta + 2 \pm \sqrt{\vartheta^2 - 4\vartheta})/(2\vartheta + 2\eta + 2)$, whose absolute values are all at most 1, which represents that the poles are in a unit circle, and thus equation (5.27) is steady. So we have

$$\lim_{k \rightarrow \infty} \epsilon^{\iota}(k) = \lim_{z \rightarrow 1} (z-1) \cdot \epsilon^{\iota}(z) = \frac{\lambda^{\iota}\tau}{\eta} = \frac{\lambda^{\iota}}{\psi}.$$

Consequently, we come to a conclusion that $\lim_{k \rightarrow \infty} \|\epsilon_k\|_2 = \|\lambda/\psi\|_2 + O(\tau^2)$, despite of how large $\mathcal{G}(t)$ is. Besides, for any constant noises, λ is always 0 and thus $\lim_{k \rightarrow \infty} \epsilon^{\iota}(k) = 0$, which explains that the residual error $\lim_{k \rightarrow \infty} \|\epsilon_k\|_2$ converges to $O(\tau^2)$ in the presence of constant noises. The proof is thus completed. $\square$

Theorem 5.4 *The residual error for solving time-dependent Sylvester equations (5.1), when the designed DCND model (5.7) is perturbed by the bounded random noise $\chi(t)$, globally converges to $2np \cdot \sup_{1 \leq i \leq k, 1 \leq t \leq np} |\chi_i^t| / (1 - \|\mathcal{L}\|_2) + O(\tau^2)$.*

Proof Similar to the proof in Theorem 5.3, the analyses can be divided into the bounded random noise $\chi(t)$ and $O(\tau^2)$. We can obtain the following equation based on (5.23):

$$(1 + \vartheta + \eta)\epsilon_{k+1}^t = (2 + \vartheta)\epsilon_k^t - \epsilon_{k-1}^t + \chi_k^t - \chi_{k-1}^t. \quad (5.29)$$

Let $\zeta_k^t = [\chi_k^t - \chi_{k-1}^t, 0]^T$. The above equation can be reformulated into a state-space equation in control as follows:

$$\rho_{k+1}^t = \mathcal{L} \cdot \rho_k^t + \zeta_k^t.$$

Then, we can obtain

$$\begin{aligned} \|\rho_{k+1}^t\|_2 &\leq \|\mathcal{L} \cdot \rho_k^t\|_2 + \|\zeta_k^t\|_2 \\ &\leq \|\mathcal{L}\|_2 \cdot \|\rho_k^t\|_2 + \|\zeta_k^t\|_2 \\ &\leq \|\mathcal{L}\|_2 \|\mathcal{L} \rho_{k-1}^t\|_2 + \|\mathcal{L}\|_2 \|\zeta_{k-1}^t\|_2 + \|\zeta_k^t\|_2 \\ &\leq \|\mathcal{L}\|_2^2 \|\rho_{k-1}^t\|_2 + \|\mathcal{L}\|_2 \|\zeta_{k-1}^t\|_2 + \|\zeta_k^t\|_2 \\ &\dots \\ &\leq \|\mathcal{L}\|_2^k \|\rho_1^t\|_2 + \|\mathcal{L}\|_2^{k-1} \|\zeta_1^t\|_2 + \dots + \|\zeta_k^t\|_2 \\ &< \|\mathcal{L}\|_2^k \|\rho_1^t\|_2 + \max_{1 \leq i \leq k} \|\zeta_i^t\|_2 / (1 - \|\mathcal{L}\|_2) \\ &< \|\mathcal{L}\|_2^k \|\rho_1^t\|_2 + 2 \max_{1 \leq i \leq k} |\chi_i^t| / (1 - \|\mathcal{L}\|_2). \end{aligned}$$

In view of $\lim_{k \rightarrow \infty} \|\mathcal{L}\|_2^k = 0$, the above formula can further reduce

$$\lim_{k \rightarrow \infty} \|\rho_{k+1}^t\|_2 < 2 \max_{1 \leq i \leq k} |\chi_i^t| / (1 - \|\mathcal{L}\|_2).$$

Then we can obtain

$$\lim_{k \rightarrow \infty} \|\epsilon_k\|_2 < 2np \sup_{1 \leq i \leq k, 1 \leq t \leq np} \frac{\chi_i^t}{1 - \|\mathcal{L}\|_2} + O(\tau^2).$$

The proof is thus completed. $\square$

In what follows, the following theorems are offered to show the convergence property of DCND-BFGS model (5.8).

Theorem 5.5 *When the approximate inverse $\tilde{E}_k$ in DCND-BFGS model (5.8) converges to Q_k^{-1} during the sampling gap τ with a residual error $\mathcal{J}(\tau^2)$, i.e., $\tilde{E}_k - Q_k^{-1} = \mathcal{J}(\tau^2)$, and DCND-BFGS model (5.8) accomplishes an update, the residual error synthesized by DCND-BFGS model (5.8) for solving time-dependent Sylvester equations (5.1) globally converges to $O(\tau^2)$, i.e., $\lim_{k \rightarrow \infty} \|\epsilon_k\|_2 = O(\tau^2)$.*

Proof In time of each sampling gap τ , we have

$$\tilde{E}_k = Q_k^{-1} + \mathcal{J}(\tau^2).$$

Then, DCND-BFGS model (5.8) can be rewritten as

$$\begin{aligned} \mathbf{y}_{k+1} = & \mathbf{y}_k + (Q_k^{-1} + \mathcal{J}(\tau^2))(-(Q_k - Q_{k-1})\mathbf{y}_k + \mathbf{a}_k - \\ & \mathbf{a}_{k-1} - \phi\tau(Q_k\mathbf{y}_k - \mathbf{a}_k) - \psi\tau \sum_{i=0}^k (Q_i\mathbf{y}_i - \mathbf{a}_i)), \end{aligned}$$

which can be further rearranged as

$$\begin{aligned} \mathbf{y}_{k+1} = & \mathbf{y}_k + Q_k^{-1}(-(Q_k - Q_{k-1})\mathbf{y}_k + \mathbf{a}_k - \mathbf{a}_{k-1} \\ & - \phi\tau(Q_k\mathbf{y}_k - \mathbf{a}_k) - \psi\tau \sum_{i=0}^k (Q_i\mathbf{y}_i - \mathbf{a}_i)) + \mathbf{O}(\tau^2), \end{aligned}$$

which can be converted into the following equation aided with Theorem 5.1:

$$(1 + \vartheta + \eta)\epsilon_k - \epsilon_{k-1} + \eta \sum_{i=0}^{k-1} \epsilon_i + \mathbf{O}(\tau^2) + \mathbf{O}(\tau^2) = 0.$$

The above equation can be simplified as

$$(1 + \vartheta + \eta)\epsilon_k - \epsilon_{k-1} + \eta \sum_{i=0}^{k-1} \epsilon_i + \mathbf{O}(\tau^2) = 0.$$

Accordingly, it can be summarized from Theorem 5.2 that the residual error synthesized by DCND-BFGS model (5.8) for solving time-dependent Sylvester equations (5.1) globally converges to $O(\tau^2)$, i.e., $\lim_{k \rightarrow \infty} \|\epsilon_k\|_2 = O(\tau^2)$. The proof is thus completed. $\square$

Theorem 5.6 According to Theorems 5.3 and 5.4 and in respect to the relationship between the given DCND-BFGS model (5.8) and DCND model (5.7) as depicted in Theorem 5.5, the residual error synthesized by DCND-BFGS model (5.8) and perturbed by the linear time-dependent noise $\mathcal{G}(t) = \lambda t + \varpi$ or bounded random noise $\chi(t)$ for solving time-dependent Sylvester equations (5.1) globally converges to $\|\lambda/\psi\|_2 + O(\tau^2)$ and $2np \cdot \sup_{1 \leq i \leq k, 1 \leq l \leq np} |\chi_i^l| / (1 - \|\mathcal{L}\|_2) + O(\tau^2)$, respectively.

Proof It can be generalized from Theorems 6.3 and 6.4 and thus omitted. $\square$

Theorem 5.7 The residual error $\|\epsilon_k\|_2$ of traditional methods, which does not utilize the velocity compensation in tracking the theoretical time-dependent solution, for solving time-dependent Sylvester equations (5.1) globally approaches to $O(\tau)$.

Proof Exploiting the time derivation, we have $d\mathbf{y}_k^l/dt = \lim_{\tau \rightarrow 0} \Delta\mathbf{y}_k^l/\tau = \dot{\mathbf{y}}_k^l$, of which $\mathbf{y}_k^l$ denotes the l th value of $\mathbf{y}$ at time instant $t = k\tau$ with $\dot{\mathbf{y}}_k^l$ being its derivation. Hence, $\Delta\mathbf{y}_k^l \approx \tau\dot{\mathbf{y}}_k^l$. For traditional methods, $\Delta\mathbf{y}_k^l = O(\tau)$ and $\Delta\mathbf{y}_k = \mathbf{O}(\tau)$ can be concluded. Further, we can get $\mathbf{y}_k^* = \mathbf{y}_{k-1}^* + \Delta\mathbf{y}_{k-1}$, which can be further rephrased as $\mathbf{y}_k^* = \mathbf{y}_{k-1}^* + \mathbf{O}(\tau) = \mathbf{y}_k + \mathbf{O}(\tau)$, where $\mathbf{y}_k^*$ and $\mathbf{y}_{k-1}^*$ signify the theoretical solution at $t = k\tau$ and $t = (k-1)\tau$, separately. In consequence,

$$\begin{aligned} \|\epsilon_k\|_2 &= \|\mathcal{Q}_k\mathbf{y}_k - \mathbf{a}_k\|_2 \\ &= \|\mathcal{Q}_k\mathbf{y}_k^* - \mathbf{a}_k + \mathcal{Q}_k\mathbf{O}(\tau)\|_2 \\ &= \|\mathcal{Q}_k\mathbf{O}(\tau)\|_2 \\ &= \|\mathbf{O}(\tau)\|_2 \\ &= O(\tau). \end{aligned}$$

The proof is thus completed. $\square$

5.4 Illustrative Examples

In this section, illustrative examples are supplied to verify the availability and superiority of the constructed models compared with the existing models.

5.4.1 Example 1

In this section, time-dependent Sylvester equations (5.1) with parameters $n = 2$ and $p = 2$ are considered for illustration and comparison with the following coefficients:

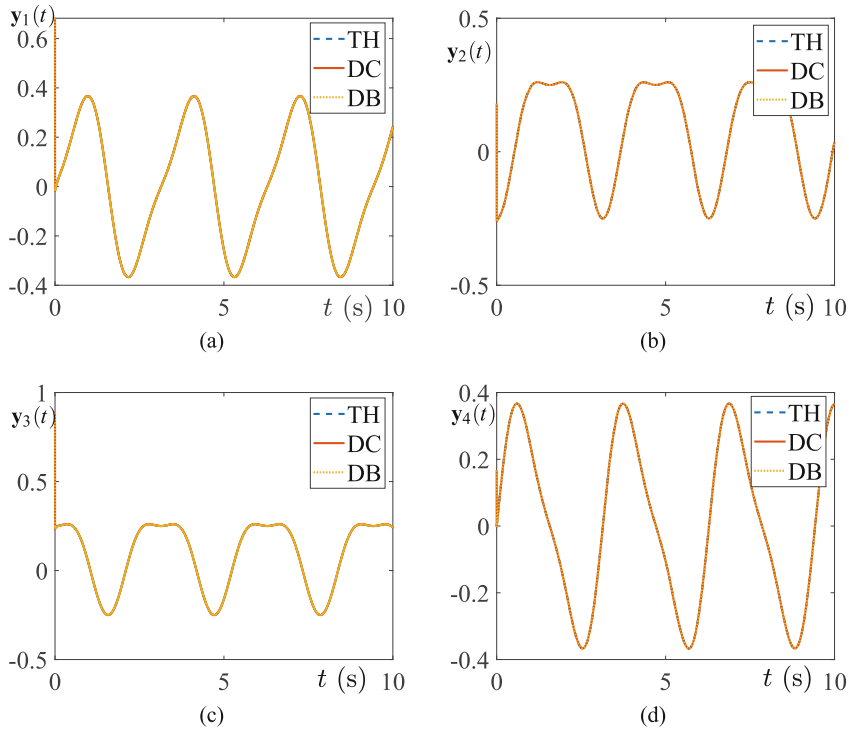


Fig. 5.1 Transient behavior of $\mathbf{y}(t)$ synthesized by the presented models for solving TDLE problem (5.30) without perturbations with $\tau = 0.001$ s and randomly generated initial values $\mathbf{y}_0 \in [0, 1]^4$, where TH corresponds to the theoretical solutions, DC corresponds to the generated solutions synthesized by the constructed DCND model (5.7), and DB corresponds to the generated solutions synthesized by DCND-BFGS model (5.8). (a) Profile of $\mathbf{y}_1(t)$, (b) profile of $\mathbf{y}_2(t)$, (c) profile of $\mathbf{y}_3(t)$, and (d) profile of $\mathbf{y}_4(t)$

$$\begin{aligned}
 M(t) &= \begin{bmatrix} -1 - 0.5\cos(2t) & 0.5\sin(2t) \\ 0.5\sin(2t) & -1 + 0.5\cos(2t) \end{bmatrix}, \\
 N(t) &= \begin{bmatrix} 1 + 0.5\cos(2t) & -0.5\sin(2t) \\ -0.5\sin(2t) & 1 - 0.5\cos(2t) \end{bmatrix}, \\
 C(t) &= \begin{bmatrix} 0.5\sin(2t) & -0.5\cos(2t) \\ 0.5\cos(2t) & 0.5\sin(2t) \end{bmatrix}.
 \end{aligned} \tag{5.30}$$

The numerical experimental results are shown in Figs. 5.1, 5.2, 5.3, 5.4, 5.5 and Table 5.1.

Firstly, to demonstrate the effectiveness of the presented DCND model (5.7) and DCND-BFGS model (5.8) for solving time-dependent Sylvester equations (5.1), the two models are used to solve problem (5.30) specifically. It can be observed

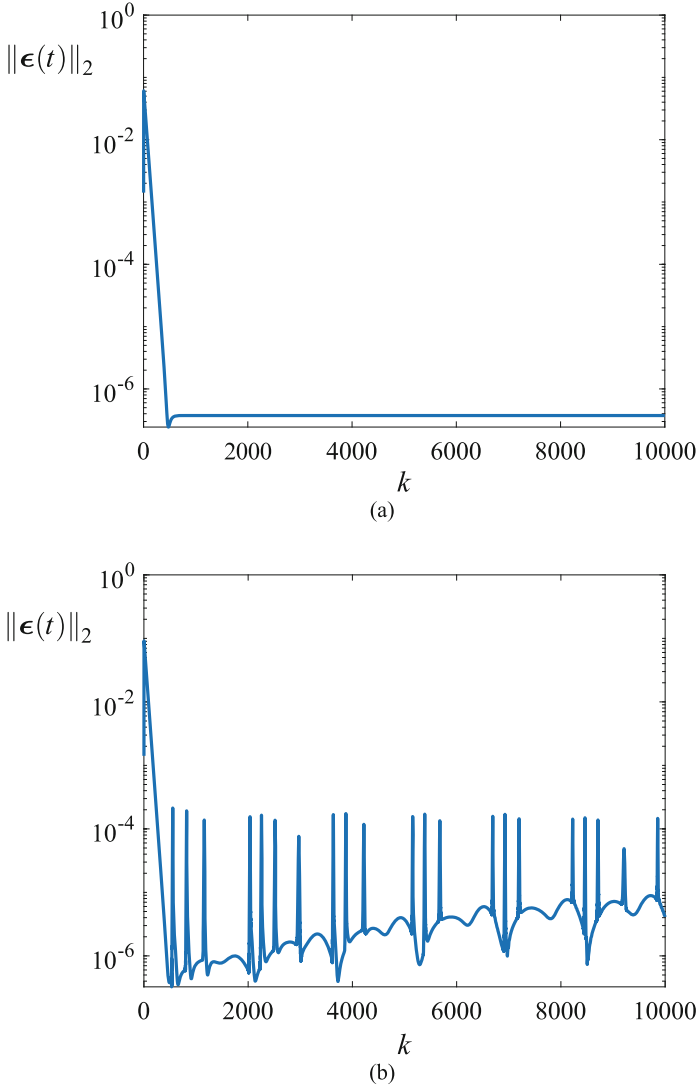


Fig. 5.2 Transient behavior of the residual errors synthesized by the designed DCND model (5.7) and DCND-BFGS model (5.8) for solving TDLE problem (5.30) without perturbations with $\tau = 0.001$ s and randomly generated initial values $\mathbf{y}_0 \in [0, 1]^4$. **(a)** Profiles of $\|\epsilon(t)\|_2$ synthesized by the given DCND model (5.7) and **(b)** profiles of $\|\epsilon(t)\|_2$ synthesized by the presented DCND-BFGS model (5.8)

from Fig. 5.1 that the generated solutions (i.e., DC and DB) all fast converge to the theoretical ones (i.e., TH) with the residual errors shown in Fig. 5.2, which indicates the two models have a similar performance. The residual error synthesized by the

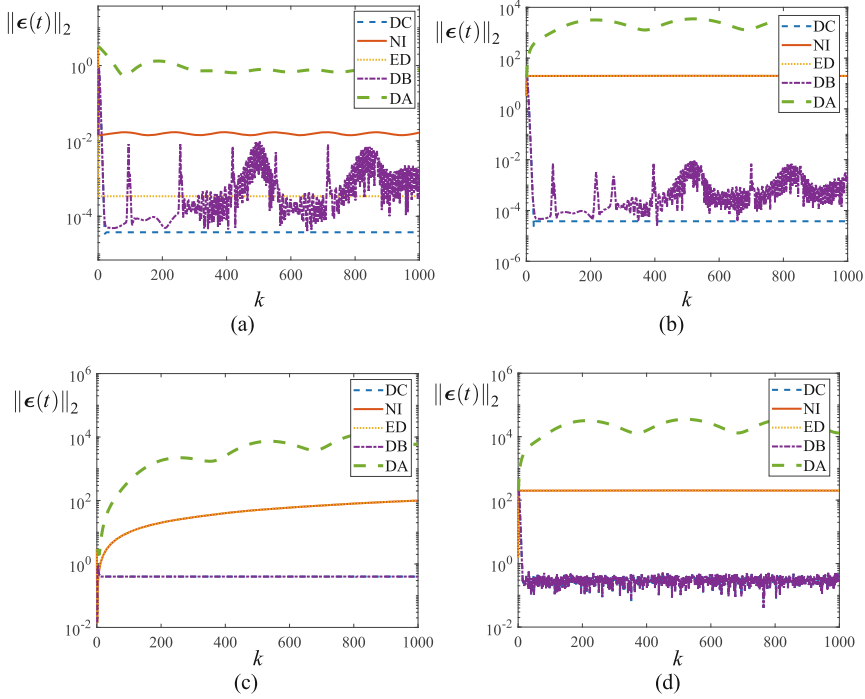


Fig. 5.3 Convergence performance of the designed DCND model (5.7) (denoted by DC), the given DCND-BFGS model (5.8) (denoted by DB), traditional EDZND model (5.10) (denoted by ED), Newton iteration (5.11) (denoted by NI), and Davidenko method (5.12) (denoted by DA) for solving TDLE problem (5.30) with $\tau = 0.01$. (a) Without noise, (b) with constant noise, (c) with linear time-dependent noise, and (d) with random noise

constructed DCND model (5.7) is shown in Fig. 5.2a and of the order 10^{-7} , while that synthesized by the designed DCND-BFGS model (5.8) is shown as Fig. 5.2b and of the order 10^{-4} , and there exists sudden protuberances of the order 10^{-4} but can rapidly restore to a smaller convergence value of the order 10^{-6} . Furthermore, it can be seen from the average computing time per updating (ACTPU) in Table 5.1 that the computing speed of DCND-BFGS model (5.8) is faster than that of DCND model (5.7), because the BFGS algorithm avoids the matrix inverse operation and then saves the operation time. The above results illustrated in Figs. 5.1 and 5.2 preliminarily validate the high efficacy and accuracy of the two models with the correctness of proof in Theorems 6.2 and 6.3.

Secondly, for further investigation, the given DCND model (5.7), the presented DCND-BFGS model (5.8), traditional EDZND model (5.10), Newton iteration (5.11), and Davidenko method (5.12) are exploited in the same problem (5.30) as shown in Figs. 5.3, 5.4, and 5.5 by taking sampling gap $\tau = 0.01$, 0.001 , and 0.0001 s and without noises, constant noises, linear time-dependent noises,

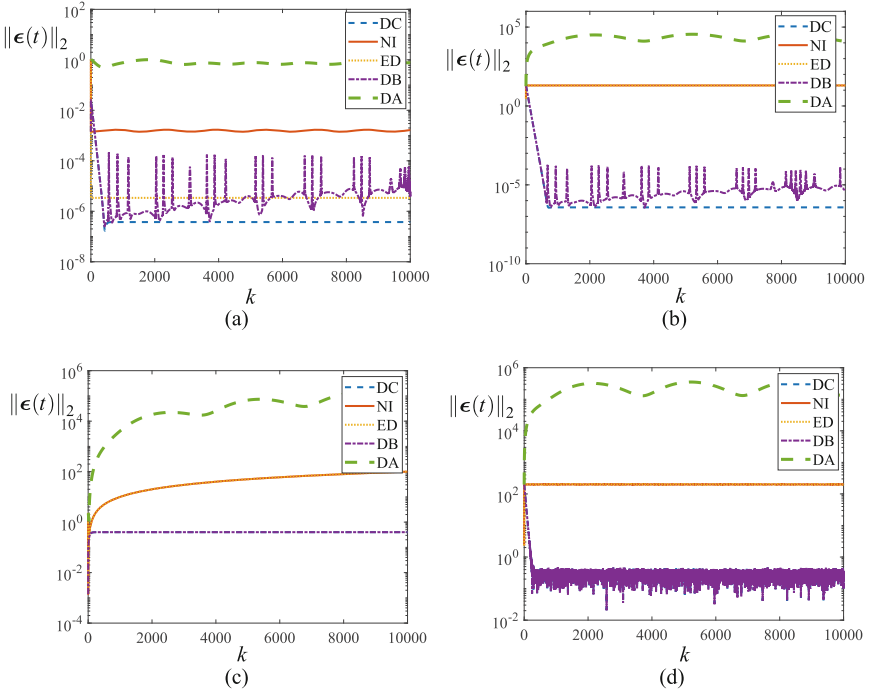


Fig. 5.4 Convergence performance of the presented DCND model (5.7) (denoted by DC), the constructed DCND-BFGS model (5.8) (denoted by DB), traditional EDZND model (5.10) (denoted by ED), Newton iteration (5.11) (denoted by NI), and Davidenko method (5.12) (denoted by DA) for solving TDLE problem (5.30) with $\tau = 0.001$. (a) Without noise, (b) with constant noise, (c) with linear time-dependent noise, and (d) with random noise

and random noises into consideration. From Fig. 5.3, it can be seen that Newton iteration (5.11) and Davidenko method (5.12) have poor performance in achieving an accurate solution as compared with the other three models. Specifically, as shown in Table 5.1 as well as Fig. 5.3a, for $\tau = 0.01$ s with the ACTPU less than the sampling gap τ , the maximal steady-state residual errors (MSSREs) synthesized by the constructed DCND model (5.7), the designed DCND-BFGS model (5.8), traditional EDZND model (5.10), and Newton iteration (5.11) are of the order 10^{-5} , 10^{-3} , 10^{-4} , and 10^{-2} , respectively, while Davidenko method (5.12) is of the order 10^{-1} , which demonstrates the superiority of the given models (5.7) and (5.8). Regarding the constant noises, Fig. 5.3b and Table 5.1 present the corresponding experimental results, which shows that the residual errors of the presented models (5.7) and (5.8) achieve rapid convergence in approaching zero with the MSSREs at the order of 10^{-5} and 10^{-3} , respectively. In contrast, the MSSREs of traditional EDZND model (5.10) and Newton iteration (5.11) in the presence of randomly generated constant noises are of the order 10^1 , which is about 10^6 and 10^4 times larger than that of the constructed models, separately.

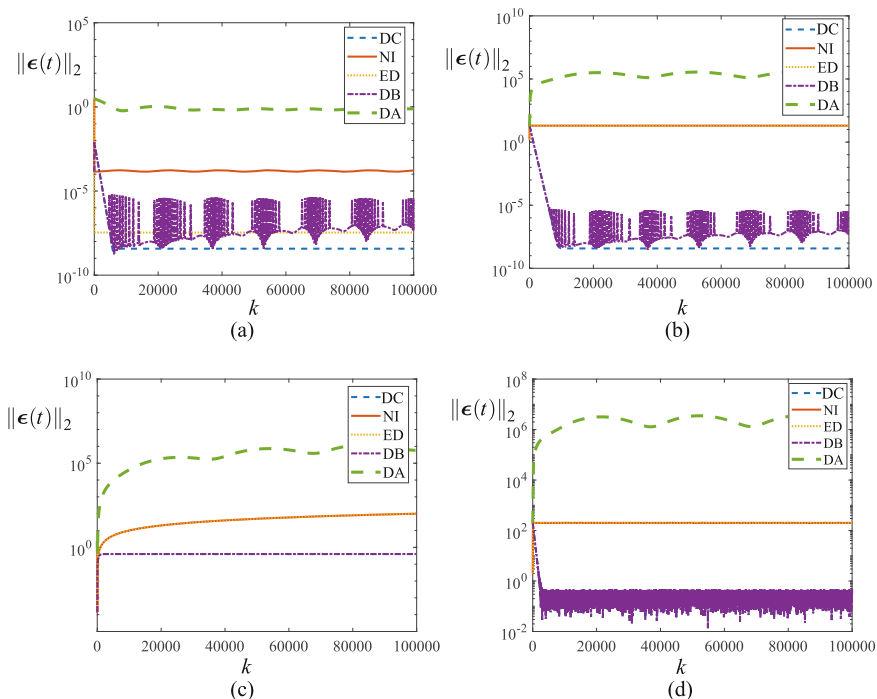


Fig. 5.5 Convergence performance of the designed DCND model (5.7) (denoted by DC), the given DCND-BFGS model (5.8) (denoted by DB), traditional EDZND model (5.10) (denoted by ED), Newton iteration (5.11) (denoted by NI), and Davidenko method (5.12) (denoted by DA) for solving TDLE problem (5.30) with $\tau = 0.0001$. (a) Without noise, (b) with constant noise, (c) with linear time-dependent noise, and (d) with random noise

Besides, Davidenko method (5.12) cannot complete the given task with an infinite residual error. As a continuation, Fig. 5.3c and d provides results on the residual errors synthesized by these models with respect to linear noises and random noises severally. It can be observed from these two figures and Table 5.1 that, for linear noises, the MSSREs of the designed models (5.7) and (5.8) maintain a relatively small convergence value while those of the rest go to infinity and that, for nonzero mean random noises, the given models (5.7) and (5.8) perform the best among these computational methods. These results further illustrate that the presented models (5.7) and (5.8) have a strong stability and can remedy the weaknesses of traditional computational methods in solving time-dependent Sylvester equations effectively.

Thirdly, compared with the residual errors in Figs. 5.3, 5.4, and 5.5 with different τ , we can see the designed DCND model (5.7), the given DCND-BFGS model (5.8), and traditional EDZND model (5.10) change in an $O(\tau^2)$ manner, while that of

Table 5.1 Comparisons among different models for solving TDLE problem (5.30) in terms of MSSRE with the average computing time per updating (ACTPU) in brackets, where their initial states are generated randomly

τ (s)	Model	MSSRE and ACTPU (s) in brackets without noises	MSSRE and ACTPU (s) in brackets with randomly generated constant noises	MSSRE and ACTPU (s) in brackets with linear noises $\mathcal{G}(t) = 5 * (t - 1)$	MSSRE and ACTPU (s) in brackets with random noises $\mathcal{G}(t) \in [99.5, 100.5]$
0.01	DCND (5.7)	3.764×10^{-5} (3.419×10^{-5})	3.764×10^{-5} (3.117×10^{-5})	4.000×10^{-1} (2.905×10^{-5})	5.041×10^{-1} (2.900×10^{-5})
	DCND-BFGS (5.8)	7.200×10^{-3} (2.678×10^{-5})	7.100×10^{-3} (2.864×10^{-5})	4.049×10^{-1} (2.646×10^{-5})	5.062×10^{-1} (2.661×10^{-5})
	EDZND (5.10)	3.404×10^{-4} (1.686×10^{-5})	2.000×10^1 (1.813×10^{-5})	Infinity	2.008×10^2 (1.822×10^{-5})
	NI (5.11)	1.700×10^{-2} (9.649×10^{-6})	2.023×10^1 (1.031×10^{-5})	Infinity	2.032×10^2 (9.579×10^{-6})
	Davidenko (5.12)	7.709×10^{-1} (1.051×10^{-5})	NA	NA	NA
0.001	DCND (5.7)	3.739×10^{-7} (2.281×10^{-5})	3.739×10^{-7} (2.265×10^{-5})	4.000×10^{-1} (2.226×10^{-5})	4.835×10^{-1} (2.260×10^{-5})
	DCND-BFGS (5.8)	1.530×10^{-4} (2.040×10^{-5})	1.440×10^{-4} (2.003×10^{-5})	4.001×10^{-1} (1.983×10^{-5})	4.836×10^{-1} (2.016×10^{-5})
	EDZND (5.10)	3.400×10^{-6} (1.299×10^{-5})	2.000×10^1 (1.273×10^{-5})	Infinity	2.009×10^2 (1.321×10^{-5})
	NI (5.11)	1.700×10^{-3} (7.235×10^{-6})	2.002×10^1 (7.142×10^{-6})	Infinity	2.011×10^2 (7.170×10^{-6})
	Davidenko (5.12)	7.679×10^{-1} (7.478×10^{-6})	NA	NA	NA
0.0001	DCND (5.7)	3.737×10^{-9} (2.190×10^{-5})	3.737×10^{-9} (2.135×10^{-5})	4.000×10^{-1} (2.163×10^{-5})	4.855×10^{-1} (2.188×10^{-5})
	DCND-BFGS (5.8)	4.065×10^{-6} (1.990×10^{-5})	3.932×10^{-6} (1.949×10^{-5})	4.000×10^{-1} (1.934×10^{-5})	4.855×10^{-1} (1.942×10^{-5})
	EDZND (5.10)	3.399×10^{-8} (1.225×10^{-5})	2.000×10^1 (1.225×10^{-5})	Infinity	2.010×10^2 (1.224×10^{-5})
	NI (5.11)	1.700×10^{-4} (6.979×10^{-6})	2.000×10^1 (7.063×10^{-6})	Infinity	2.010×10^2 (6.846×10^{-6})
	Davidenko (5.12)	7.681×10^{-1} (6.778×10^{-6})	NA	NA	NA

“NA” means that Davidenko method (5.12) cannot correctly obtain $\mathbf{y}(t)$ within a limited and relatively small error range

Newton iteration (5.11) changes in an $O(\tau)$ manner, without perturbations or with constant noises. These results further validate the correctness of the proofs in Theorems 5.2, 5.5 and 5.7. Moreover, for linear or random noises, the MSSREs of the presented DCND model (5.7) and the constructed DCND-BFGS model (5.8)

remain at a certain value, which demonstrates the correctness of Theorems 5.3, 5.4, and 5.6, while traditional EDZND model (5.10), Newton iteration (5.11), and Davidenko Method (5.12) fail to handle these noises with nonnegligible residual errors.

5.4.2 Example 2: Tracking Control of Robotics

In this section, the designed DCND model (5.7) is exploited to the motion tracking of redundant robot manipulators. We consider the case of tracking a butterfly curve with an UR5 robot specifically, which is reported in [28], and other redundant robot manipulators can be similarly derived and omitted here [29–32]. The affine system at the joint-velocity level originating from the forward kinematics equation of UR5 robot is defined as

$$J(\theta(t))\dot{\theta}(t) = \dot{\varphi}(t), \quad (5.31)$$

where $J(\theta(t))$ is the Jacobian matrix with $J(\theta(t)) \in \mathbb{R}^{3 \times 6}$; $\theta(t) \in \mathbb{R}^6$ is the joint angle vector of the UR5 robot; $\varphi(t) \in \mathbb{R}^3$ is the end-effector position with $\dot{\varphi}(t)$ being its time derivative. For this system, the simulation results synthesized by the given DCND model (5.7) are shown in Fig. 5.6 with the corresponding parameters being set as the execution time T of 10 s and $\tau = 0.001$ s. The robot manipulators execute the butterfly curve successfully simulated on the V-rep simulation platform as Fig. 5.6a shows and can complete perfectly with the position error $\varepsilon = \varphi(t) - \varphi_d(t)$ being of the order 10^{-4} m in the case of injecting random noises $\mathcal{G}(t) \in [-1, 1] + 5$ as shown in Fig. 5.6d, where $\varphi_d(t)$ describes the desired trajectory, which embodies the high effectiveness and stability of the manipulator to execute tasks synthesized by the presented DCND model (5.7).

5.4.3 Example 3: Linear Detection for MIMO

In this section, we consider a massive multiple-input multiple-output (MIMO) system, which is one of the most promising technologies in communication fields, like 5G [33, 34], and can improve data rate and reliability eminently. Considering the uplink data detection, we give the definitions as the following where the transmitted signal and received signal are presented by vector $\mathbf{P} = (p_1, p_2, \dots, p_{N_t})^T \in \mathbb{C}^{N_t \times 1}$ and $\mathbf{Y} = (\gamma_1, \gamma_2, \dots, \gamma_{N_r})^T \in \mathbb{C}^{N_r \times 1}$, where N_t and N_r denote the transmitted antennas and receive antennas (users) in the base station, respectively, that $\mathbf{H} = (\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_{N_t}) \in \mathbb{C}^{N_r \times N_t}$ with $\mathbf{h}_k = (h_{1,k}, h_{2,k}, \dots, h_{N_r,k}) \in \mathbb{C}^{N_r \times 1}$ and $h_{i,k}$ means the complex channel gain from transmit antenna k to receive antenna i , and that $\mathbf{\mathcal{L}} = (\mathcal{L}_1, \mathcal{L}_2, \dots, \mathcal{L}_{N_r})^T \in \mathbb{C}^{N_r \times 1}$ signifies the white Gaussian noise vector whose elements are mutually independent and zero mean with a variance N_0 .

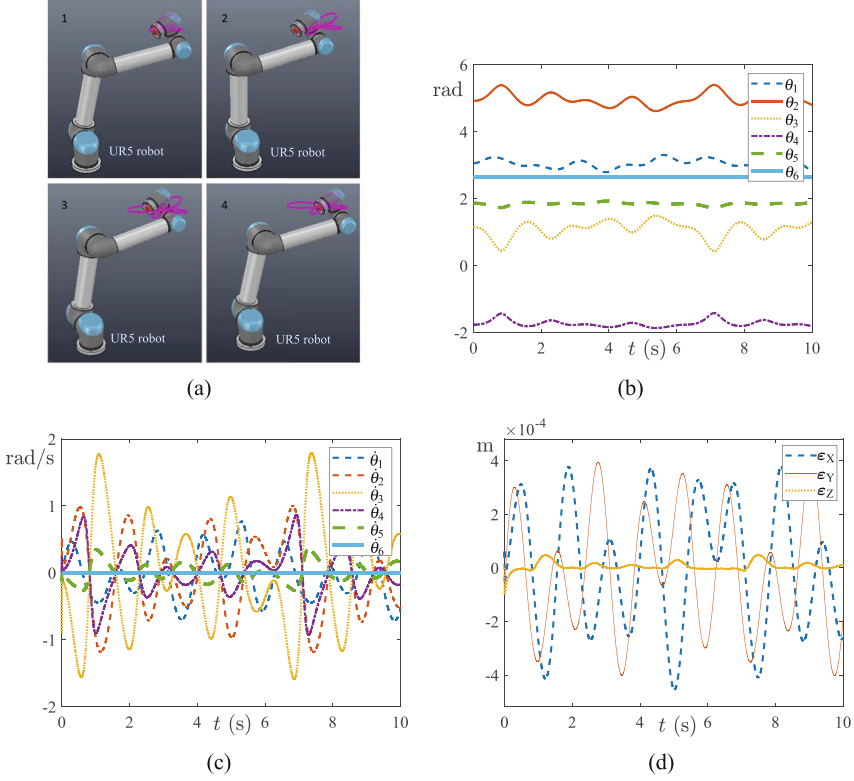


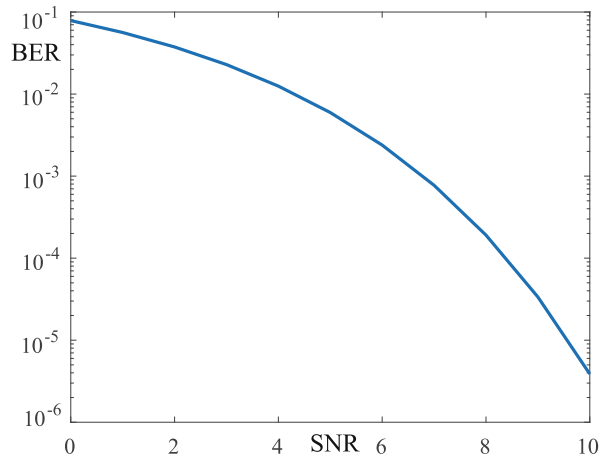
Fig. 5.6 Simulation results of the UR5 robot tracking the butterfly curve synthesized by the constructed DCND model (5.7) with $\tau=0.001$ s. (a) Profiles of motion trajectory in four different moments, (b) profiles of joint angle, (c) profiles of joint velocity, and (d) profiles of position error

Thus, we can design a communication system as

$$\mathbf{Y} = H\mathbf{p} + \boldsymbol{\varepsilon}. \quad (5.32)$$

To obtain the estimated solution $\hat{\mathbf{p}}$ with the known $\mathbf{Y}$, matrix inversion operation is involved inevitably. Therefore, the designed simple communication system is used to estimate the transmitted signal and determine the performance of the communication channels which is characterized by bit error rate (BER) tested by the method of direct matrix inversion and the designed DCND model (5.7). It can be observed from Fig. 5.7 that a higher ratio of N_r/N_t leads to a better BER synthesized by the given models. When the value of N_r/N_t is high, the BER can be lower with the order of 10^{-5} at a very low signal noise ratio (SNR). As a conclusion, the presented DCND model (5.7) has a favorable BER performance and can be preliminarily applied to the MIMO domain as a bold attempt.

Fig. 5.7 BER performance for data detection with the configuration being set that the number of transmitted antennas is 2, and the ratio of transmitted antennas to the user number N_r/N_t is 1 synthesized by the constructed DCND model (5.7)



5.5 Conclusions

In this chapter, two discrete computational models (DCND and DCND-BFGS) have been constructed for solving time-dependent Sylvester equations, and their corresponding global convergence performance and anti-perturbation property have been derived theoretically. For comparative purposes, traditional EDZND model (5.10), Newton iteration (5.11), and Davidenko method (5.12) have been introduced for solving the same problems. The illustrative examples and practical applications in control and communication domains have further verified the superiority and high efficacy of the designed models.

References

1. Wei Q, Dobigeon N, Tournet JY, Bioucas-Dias J, Godsill S (2016) R-FUSE: robust fast fusion of multiband images based on solving a Sylvester equation. *IEEE Signal Process Lett* 23(11):1632–1636
2. Lee HC, Choi JW (2004) Linear time-varying eigenstructure assignment with flight control application. *IEEE Trans Aerospace Electron Syst* 40(1):145–157
3. Chen G, Song Y, Wang F, Zhang C (2008) Semi-supervised multi-label learning by solving a Sylvester equation. In: *Proceedings of the 2008 SIAM international conference on data mining* pp 410–419
4. Harker M, O’Leary P (2011) Least squares surface reconstruction from gradients: direct algebraic methods with spectral, Tikhonov, and constrained regularization. In: *IEEE/CVF conference on computer vision and pattern recognition*, pp 2529–2536
5. Wang L, Li D, He T, Xue Z (2016) Manifold regularized multi-view subspace clustering for image representation. In: *International conference on pattern recognition*, pp 283–288
6. Shaker HR, Tahavori M (2015) Control configuration selection for bilinear systems via generalised Hankel interaction index array. *Int J Control* 88(1):30–37

7. Bartels RH, Stewart GW (1972) Solution of the matrix equation $AX + XB = C$. *Commun ACM* 15(9):820–826
8. Golub G, Nash S, Van Loan C (1979) A Hessenberg-Schur method for the problem $AX + XB = C$. *IEEE Trans Autom Control* 24(6):909–913
9. Jin L, Zhao J, Chen L, Li S (2024) Collective neural dynamics for sparse motion planning of redundant manipulators without Hessian matrix inversion. *IEEE Trans Neural Netw Learn Syst*. In Press. <https://doi.org/10.1109/TNNLS.2024.3363241>
10. Wei L, Jin L (2024) Collaborative neural solution for time-varying nonconvex optimization with noise rejection. *IEEE Trans Emerg Top Comput Intell*. In Press. <https://doi.org/10.1109/TETCI.2024.3369482>
11. Ding F, Chen T (2005) Gradient based iterative algorithms for solving a class of matrix equations. *IEEE Trans Autom Control* 50(8):1216–1221
12. Zhou B, Li ZY, Duan GR, Wang Y (2009) Weighted least squares solutions to general coupled Sylvester matrix equations. *J Comput Appl Math* 224(2):759–776
13. Song C, Feng JE, Wang X, Zhao J (2014) Finite iterative method for solving coupled Sylvester-transpose matrix equations. *J Appl Math Comput* 46(1–2):351–372
14. Ma W, Zhang Y, Wang J (2008) MATLAB simulink modeling and simulation of Zhang neural networks for online time-varying Sylvester equation solving. In: *IEEE international joint conference on neural networks*, pp 285–289
15. 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
16. Li S, Chen S, Liu B (2013) Accelerating a recurrent neural network to finite-time convergence for solving time-varying Sylvester equation by using a sign-bi-power activation function. *Neural Process Lett* 37:189–205
17. Zhang Y, Chen K, Li X, Yi C, Zhu H (2008) Simulink modeling and comparison of Zhang neural networks and gradient neural networks for time-varying Lyapunov equation solving. *Int Conf Nat Comput* 3:521–525
18. Xiao L, Zhang Y (2014) Solving time-varying inverse kinematics problem of wheeled mobile manipulators using Zhang neural network with exponential convergence. *Nonlinear Dyn* 76:1543–1559
19. Jin L, Zhang Y, Li S, Zhang Y (2016) Modified ZNN for time-varying quadratic programming with inherent tolerance to noises and its application to kinematic redundancy resolution of robot manipulators. *IEEE Trans Ind Electron* 63(11):6978–6988
20. Li S, Li Y (2013) Nonlinearly activated neural network for solving time-varying complex Sylvester equation. *IEEE Trans Cybern* 44(8):1397–1407
21. Mao M, Li J, Jin L, Li S, Zhang Y (2016) Enhanced discrete-time Zhang neural network for time-variant matrix inversion in the presence of bias noises. *Neurocomputing* 207:220–230
22. Talisa SH (1985) Application of Davidenko's method to the solution of dispersion relations in lossy waveguiding systems. *IEEE Trans Microwave Theory Tech* 33(10):967–971
23. Horn RA, Johnson CR (2012) *Matrix analysis*. Cambridge University Press, Cambridge
24. Hu JL, Wu Z, McCann H, Davis LE, Xie CG (2006) BFGS quasi-Newton method for solving electromagnetic inverse problems. *IEE Proc-Microwaves Antennas Propag* 153(2):199–204
25. Leithead WE, Zhang Y (2007) $O(N^2)$ -operation approximation of covariance matrix inverse in Gaussian process regression based on quasi-Newton BFGS method. *Commun Stat Simul Comput* 36(2):367–380
26. 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*, pp 3711–3716
27. Ding S, Yang M, Mao M, Xiao L, Zhang Y (2016) Complex ZNN and GNN models for time-varying complex quadratic programming subject to equality constraints. In: *World congress on intelligent control and automation*, pp 210–215
28. Kebria PM, Al-Wais S, Abdi H, Nahavandi S (2016) Kinematic and dynamic modelling of UR5 manipulator. In: *IEEE international conference on systems, man, and cybernetics*, pp 004229–004234

29. Xiao L, Liao B, Li S, Zhang Z, Ding L, Jin L (2017) Design and analysis of FTZNN applied to real-time solution of nonstationary Lyapunov equation and tracking control of wheeled mobile manipulator. *IEEE Trans Ind Inform* 14(1):98–105
30. Zhang Y, Li S, Gui J, Luo X (2017) Velocity-level control with compliance to acceleration-level constraints: A novel scheme for manipulator redundancy resolution. *IEEE Trans Ind Inform* 14(3):921–930
31. Jin L, Li S, Luo X, Li Y, Qin B (2018) Neural dynamics for cooperative control of redundant robot manipulators. *IEEE Trans Ind Inform* 14(9):3812–3821
32. Yang C, Jiang Y, Li Z, He W, Su CY (2016) Neural control of bimanual robots with guaranteed global stability and motion precision. *IEEE Trans Ind Inform* 13(3):1162–1171
33. Lee BM, Yang H (2017) Massive MIMO for industrial internet of things in cyber-physical systems. *IEEE Trans Ind Inform* 14(6):2641–2652
34. Lee BM (2018) Energy efficient selected mapping schemes based on antenna grouping for industrial massive MIMO-OFDM antenna systems. *IEEE Trans Ind Inform* 14(11):4804–4814

Chapter 6

High-Order Robust Discrete-Time Neural Dynamics



6.1 Introduction

Tensor problems have attracted much attention of many researchers because of their wide applications in industrial intelligence [1], multiple classification on big data [2], secure industrial Internet of Things [3], industrial Internet of Things [4, 5], image processing [6, 7], and so on [8] in recent years. For example, tensors play important roles in data analyses [9], in which they are viewed as a container for storing various kinds of data, providing great convenience for solving problems. In addition, the tensor decomposition can be found in [10, 11], which is used as a technology to achieve the feature extraction and big data processing, respectively. Furthermore, tensor compensation is utilized to make images more clear, natural, and smooth, leading to higher accuracy and less computing time for face recognition in [12].

Tensor equations have widespread applications in numerical partial differential equations and data mining, and many discrete solutions are presented and investigated to find solutions to tensor equations. For instance, the classical tensor methods in [13, 14] are extended to solve the multi-linear tensor equation (MTE) with a symmetric tensor, and a tensor method is presented based on the rank-1 approximation of the coefficient tensor in [15]. Besides, two Newton-type methods activated by traditional methods for obtaining solutions to linear equations with matrix-form coefficients are presented to solve the MTE with $\mathcal{M}$ -tensor in [16, 17]. Furthermore, a tensor splitting method is designed to solve tensor equations from the perspective of splitting modes of tensors combined with the Newton's method in [18]. Different from the method in [18], Liu et al. present a tensor splitting (TS) method by using the equivalent conditions derived from tensor splitting to solve tensor equations [19]. Then, two Levenberg-Marquardt-type methods are designed in [20, 21] to solve the MTE with the coefficient tensor being semi-symmetric. In addition, the MTE with $\mathcal{M}$ -tensor is transformed into a nonlinear optimization problem without constraints, which is solved by the presented sufficient descent

nonlinear conjugate gradient (SDNCG) method in [22]. Besides, several methods based on neural networks are designed to solve the MTE. For example, in [23], aiming at the MTE with $\mathcal{M}$ -tensor, a continuous-time neural network model with its discrete form called discrete-time neural network (DTNN) model is presented, and gradient neural network models are combined with different activation functions to generate two new gradient neural network models with their discrete forms derived by using the Euler difference formula in [24]. In addition, for MTE, Huang et al. combine Newton's method with a neural algorithm and present a modified Newton integration (MNI) neural algorithm in [25]. The above works provide efficient methods to find the solution of an MTE with $\mathcal{M}$ -tensor. However, these methods only consider the conditions with static problems, whereas most problems in practice are time-varying, which may lead to large lagging errors.

In recent years, neural dynamics (ND) methods have been frequently used to solve dynamic problems and achieved remarkable achievements [26–29]. In this chapter, an ND method is given to solve the time-varying multi-linear tensor equation (TMTE). It is readily found that the existing discrete-time ND models for solving MTE problems all rely on continuous-time models, which may not be effective for problems that are essentially discrete and are all constructed with the aid of the Euler difference formula whose solution accuracy is not high enough. In view of this, it makes sense to design a method that can efficiently solve the TMTE with $\mathcal{M}$ -tensor in real time. Therefore, for the problem on solution accuracy, the Taylor-type difference formula is used, which has higher accuracy than the Euler difference formula [30]. For the condition that discrete models based on continuous models may not be able to solve essentially discrete problems, a directly discrete method is leveraged to design an ND method for the TMTE with $\mathcal{M}$ -tensor, which is consistent with problems that are essentially discrete. Then, considering the presence of noises in the solution process, a sum term is incorporated into the design scheme. As a consequence, a high-order robust discrete-time ND (HRDND) method is proposed to solve the TMTE with $\mathcal{M}$ -tensor in this chapter. Furthermore, the proposed HRDND method is compared with some existing methods from several aspects, which are summarized in Table 6.1.

The remainder of this chapter is organized as follows. The continuous and discrete forms of the TMTE with $\mathcal{M}$ -tensor are provided in Sect. 6.2. In addition, Sect. 6.2 includes the design details of the HRDND method and shows other four state-of-the-art (SOTA) methods. In Sect. 6.3, the theoretical analyses on convergence and robustness are given. Several numerical examples solved by the HRDND method, the TS method, the SDNCG method, the DTNN model, and the MNI algorithm are provided in Sect. 6.4. Finally, Sect. 6.5 concludes this chapter. Before ending this part, we summarize contributions of this chapter as follows:

- (1) To improve the solution accuracy and adapt to the problem which is discrete in nature, an ND method is designed for solving the TMTE with $\mathcal{M}$ -tensor by exploiting the Taylor type difference formula and using a directly discrete method that is no longer dependent on a continuous model.

Table 6.1 Comparisons among some existing methods

Method	Design perspective	Derivative information†	Problem formula	Design approach	Robust
HRDND method (6.6)	Neural dynamics	Involved	Time-varying	Direct discretization	✓
New tensor method based on the rank-1 approximation [15]	Tensor method and rank-1 approximation	Not involved	Static	Direct discretization	✗
Newton method [16]	Optimization problem	Not involved	Static	Direct discretization	✗
Quadratically convergent algorithm [17]	System of nonlinear equations	Not involved	Static	Direct discretization	✗
Newton-Gauss-Seidel method [18]	Tensor splitting and Newton method	Not involved	Static	Direct discretization	✗
Tensor splitting method (6.7) [19]	Tensor splitting	Not involved	Static	Direct discretization	✗
LM method [20]	LM algorithm and optimization problem	Not involved	Static	Direct discretization	✗
TALM method [21]	LM algorithm and optimization problem	Not involved	Static	Direct discretization	✗
SDNCG method (6.8) [22]	Optimization problem	Not involved	Static	Direct discretization	✗
DTNN model (6.9) [23]	Neural dynamics	Not involved	Static	Indirect discretization	✗
DNGNN model [24]	Neural dynamics and activation function	Not involved	Static	Indirect discretization	✗
DLGNN-I model [24]	Neural dynamics	Not involved	Static	Indirect discretization	✗
DLGNN-II model [24]	Neural dynamics and preconditioning	Not involved	Static	Indirect discretization	✗
MNI model (6.10) [25]	Neural dynamics and Newton method	Not involved	Static	Indirect discretization	✓

† Denoting whether a method includes $\dot{\mathcal{R}}_k$ or $\dot{\mathbf{p}}_k$

- (2) Considering that the presence of noises is inevitable, a design formula including a sum term is provided to improve the robustness of the system. Besides, corresponding theoretical analyses of the given method and simulation results of time-varying numerical examples are provided.

6.2 Problem Formulation and Solutions

In this section, the concrete form of the TMTE with $\mathcal{M}$ -tensor is provided, and its discrete form is explicitly given. In addition, the HRDND method is constructed to solve the TMTE with $\mathcal{M}$ -tensor, and four SOTA discrete-time methods are provided for subsequent Table 6.1. Furthermore, definitions of some variables used in this chapter are given in Table 6.2.

6.2.1 Problem Formulation

For the TMTE, some preliminaries are given:

Table 6.2 Definitions of main variables

Variable	Definition
$\mathcal{R}$	An $\mathcal{M}$ -tensor
$\hat{\mathcal{R}}$	The semi-symmetric tensor of $\mathcal{R}$
$\dot{\mathcal{R}}$	The time derivative of $\mathcal{R}$
χ	The objective vector
$\dot{\chi}$	The time derivative of χ
$\mathbf{p}$	A positive vector
$\dot{\mathbf{p}}$	The time derivative of $\mathbf{p}$
α	The order of tensor $\mathcal{R}$
β	The dimension of tensor $\mathcal{R}$
k	The updating index of discrete problem
δ	The sampling interval
h	The h th row of a matrix
l	The l th column of a matrix
$\boldsymbol{\varepsilon}$	Error function vector
$\dot{\boldsymbol{\varepsilon}}$	The time derivative of $\boldsymbol{\varepsilon}$
$\mathbf{O}$	Steady-error vector
θ_1	A coefficient scalar
θ_2	A coefficient scalar
Ω	A collection
$\ \cdot\ _2$	Euclidean norm of a vector

Definition 6.1 Giving a tensor $\mathcal{R} \in \mathbb{R}^{[\alpha, \beta]}$, a scalar $u \in \mathbb{C}^\beta$, and a nonzero vector $\chi \in \mathbb{C}^\beta$, if they satisfy $\mathcal{R}\chi^{\alpha-1} = u\chi^{[\alpha-1]}$, then u is the eigenvalue and χ is the eigenvector of the tensor $\mathcal{R}$, where α and β are both positive integers [16].

Definition 6.2 Let $\rho(\mathcal{R})$ denote the spectral radius of tensor $\mathcal{R}$ with $\rho(\mathcal{R}) = \max\{|u|\}$. Then, $\mathcal{R}$ is an $\mathcal{M}$ -tensor if the tensor $\mathcal{R}$ can be written as $\mathcal{R} = z\mathcal{I} - \mathcal{D}$, in which $z \geq \rho(\mathcal{R})$, $\mathcal{I} \in \mathbb{R}^{[\alpha, \beta]}$ is an identity tensor, and $\mathcal{D} \in \mathbb{R}^{[\alpha, \beta]}$ is non-negative [16].

Based on the above preparations, the TMTE with $\mathcal{M}$ -tensor is described as

$$\mathcal{R}(t)\chi^{\alpha-1}(t) = \mathbf{p}(t),$$

where $\mathcal{R}(t) \in \mathbb{R}^{[\alpha, \beta]}$ is an $\mathcal{M}$ -tensor; $\chi(t)$ consists of $\chi_1, \chi_2, \dots, \chi_\beta$; $\mathbf{p}(t)$ is made up of $p_1, p_2, \dots, p_\beta$. Notice that $\chi(t)$ and $\mathbf{p}(t)$ are an unknown vector and a known positive one, respectively. Considering the fact that the continuous-time problem is difficult to implement on digital computers, deriving the discrete form of the TMTE (DFTMTE) with $\mathcal{M}$ -tensor is necessary, which is formulated as follows:

$$\mathcal{R}_k \chi_k^{\alpha-1} = \mathbf{p}_k, \quad (6.1)$$

where $\mathcal{R}_k$, $\chi_k^{\alpha-1}$, and $\mathbf{p}_k$ denote the value of $\mathcal{R}(t)$, $\chi^{\alpha-1}(t)$, and $\mathbf{p}(t)$ at time instant $t = k\delta$ with $\delta > 0$ denoting the sampling gap, respectively. Aiming at the DFTMTE with $\mathcal{M}$ -tensor (6.1), two relevant definitions are provided for subsequent operations.

Definition 6.3 $\mathcal{R}_k \chi_k^{\alpha-1}$ is a vector with β elements at the time instant $t = k\delta$, whose h -element is defined as follows:

$$(\mathcal{R}_k \chi_k^{\alpha-1})_h = \sum_{h_2, \dots, h_\alpha=1}^{\beta} r_{hh_2 \dots h_\alpha} \chi_{h_2} \cdots \chi_{h_\alpha},$$

where h is an integer from 1 to β ; $r_{hh_2 \dots h_\alpha}$ and χ_{h_i} with $i = 2, \dots, \alpha$ are the elements of $\mathcal{R}_k$ and χ_k , respectively.

Definition 6.4 $\hat{\mathcal{R}}_k \chi_k^{\alpha-2}$ is a matrix with β -order at the time instant $t = k\delta$, whose hl -element is defined as

$$(\hat{\mathcal{R}}_k \chi_k^{\alpha-2})_{hl} = \sum_{h_3, \dots, h_\alpha=1}^{\beta} r_{hlh_3 \dots h_\alpha} \chi_{h_3} \cdots \chi_{h_\alpha},$$

where $\hat{\mathcal{R}}_k$ is the semi-symmetric tensor of $\mathcal{R}_k$, which can be found in [25]; h and l are both integers from 1 to β ; $r_{hlh_3 \dots h_\alpha}$ denotes the element of $\hat{\mathcal{R}}_k$; χ_{h_i} with $i = 3, \dots, \alpha$ has the same meaning as in Definition 6.3.

Then, the error function of the DFTMTE with $\mathcal{M}$ -tensor at $t = k\delta$ is designed as follows for model construction:

$$\mathbf{e}_k = \mathcal{R}_k \chi_k^{\alpha-1} - \mathbf{p}_k, \quad (6.2)$$

in which $\mathbf{e}_k \in \mathbb{R}^\beta$. Evidently, the value of χ_k gets to the theoretical solution χ^* as the error $\mathbf{e}_k$ gets to $\mathbf{0}$.

6.2.2 HRDND Method

In this section, the HRDND method is proposed directly for solving the DFTMTE with $\mathcal{M}$ -tensor, without depending on the continuous-time model. Based on previous preliminaries, the error function of the DFTMTE with $\mathcal{M}$ -tensor at $t = (k+1)\delta$ is described as

$$\mathbf{e}_{k+1} = \mathcal{R}_{k+1} \chi_{k+1}^{\alpha-1} - \mathbf{p}_{k+1}.$$

Afterwards, $\mathbf{e}_{k+1}$ can be rewritten as follows according to the Taylor expansion:

$$\mathbf{e}_{k+1} = \mathbf{e}_k + \delta \dot{\mathbf{e}}_k + \mathbf{O}(\delta^2), \quad (6.3)$$

where $\mathbf{O}(\delta^2) \in \mathbb{R}^\beta$ is a vector, whose elements are all $\mathcal{O}(\delta^2)$. According to the derivative algorithm, we have

$$\dot{\mathbf{e}}_k = \frac{d\mathbf{e}_k}{dt_k} = \frac{\partial \mathbf{e}_k}{\partial t_k} + \frac{\partial \mathbf{e}_k}{\partial \chi_k} \frac{d\chi_k}{dt_k}, \quad (6.4)$$

where t_k denotes the time instant $t = k\delta$. From [16], we know $\partial \mathbf{e}_k / \partial \chi_k = \partial (\mathcal{R}_k \chi_k^{\alpha-1} - \mathbf{p}_k) / \partial \chi_k = (\alpha - 1) \hat{\mathcal{R}}_k \chi_k^{\alpha-2}$. Then, combining (6.3) and (6.4), $\mathbf{e}_{k+1}$ can be redescribed as

$$\begin{aligned} \mathbf{e}_{k+1} &= \mathbf{e}_k + \delta \dot{\mathbf{e}}_k + \mathbf{O}(\delta^2) \\ &= \mathbf{e}_k + \delta \left(\frac{\partial \mathbf{e}_k}{\partial t_k} + \frac{\partial \mathbf{e}_k}{\partial \chi_k} \frac{d\chi_k}{dt_k} \right) + \mathbf{O}(\delta^2) \\ &= \mathbf{e}_k + \delta (\hat{\mathcal{R}}_k \chi_k^{\alpha-1} - \dot{\mathbf{p}}_k + (\alpha - 1) \hat{\mathcal{R}}_k \chi_k^{\alpha-2} \dot{\chi}_k) + \mathbf{O}(\delta^2). \end{aligned}$$

From the above equation, we can get

$$\dot{\chi}_k = \frac{(\hat{\mathcal{R}}_k \chi_k^{\alpha-2})^{-1}}{\alpha - 1} \left(\frac{\mathbf{e}_{k+1} - \mathbf{e}_k}{\delta} - \hat{\mathcal{R}}_k \chi_k^{\alpha-1} + \dot{\mathbf{p}}_k \right) + \mathbf{O}(\delta^2). \quad (6.5)$$

Replacing Taylor-type three-step forward difference formula $\dot{\mathbf{x}}_k = (2\mathbf{x}_{k+1} - 3\mathbf{x}_k + 2\mathbf{x}_{k-1} - \mathbf{x}_{k-2})/2\delta$ with $\dot{\mathbf{x}}_k$ in (6.5), the foresaid equation is rewritten as

$$\begin{aligned}\dot{\mathbf{x}}_k &= \frac{2\mathbf{x}_{k+1} - 3\mathbf{x}_k + 2\mathbf{x}_{k-1} - \mathbf{x}_{k-2}}{2\delta} \\ &= \frac{(\hat{\mathcal{R}}_k \mathbf{x}_k^{\alpha-2})^{-1}}{\alpha - 1} \left(\frac{\mathbf{e}_{k+1} - \mathbf{e}_k}{\delta} - \dot{\mathcal{R}}_k \mathbf{x}_k^{\alpha-1} + \dot{\mathbf{p}}_k \right) + \mathcal{O}(\delta^2).\end{aligned}$$

Define $\mathbf{e}_{k+1} = \theta_1 \mathbf{e}_k - \theta_2 \sum_{i=0}^k \mathbf{e}_i$ when $k \rightarrow \infty$. Then, the following formula is evidently obtained:

$$\begin{aligned}\mathbf{x}_{k+1} &= \frac{3}{2}\mathbf{x}_k - \mathbf{x}_{k-1} + \frac{1}{2}\mathbf{x}_{k-2} - \frac{(\hat{\mathcal{R}}_k \mathbf{x}_k^{\alpha-2})^{-1}}{\alpha - 1} ((1 - \theta_1)\mathbf{e}_k \\ &\quad + \theta_2 \sum_{i=0}^k \mathbf{e}_i + \delta \dot{\mathcal{R}}_k \mathbf{x}_k^{\alpha-1} - \delta \dot{\mathbf{p}}_k) + \mathcal{O}(\delta^3),\end{aligned}$$

where $\theta_1 > 0$ and $\theta_2 > 0$ are parameters of the discrete-time multi-linear system. Then, ignoring the truncation $\mathcal{O}(\delta^3)$ term, the HRDND method for solving the DFTMTE with $\mathcal{M}$ -tensor is obtained as

$$\begin{aligned}\mathbf{x}_{k+1} &= \frac{3}{2}\mathbf{x}_k - \mathbf{x}_{k-1} + \frac{1}{2}\mathbf{x}_{k-2} - \frac{1}{\alpha - 1} (\hat{\mathcal{R}}_k \mathbf{x}_k^{\alpha-2})^{-1} \\ &\quad ((1 - \theta_1)(\mathcal{R}_k \mathbf{x}_k^{\alpha-1} - \mathbf{p}_k) + \theta_2 \sum_{i=0}^k (\mathcal{R}_i \mathbf{x}_i^{\alpha-1} - \mathbf{p}_i) \\ &\quad + \delta \dot{\mathcal{R}}_k \mathbf{x}_k^{\alpha-1} - \delta \dot{\mathbf{p}}_k).\end{aligned}\tag{6.6}$$

6.2.3 SOTA Methods

For subsequent Table 6.1 to highlight the superiority of the proposed HRDND method (6.6), other four SOTA discrete-time solutions to DFTMTE with $\mathcal{M}$ -tensor (6.1) are given in this section.

(1) The TS method [19], which is designed to solve (6.1) is given as

$$\mathbf{x}_{k+1} = (M(\mathcal{E}_k)^{-1} \mathcal{F}_k \mathbf{x}_k^{\alpha-1} + M(\mathcal{E}_k)^{-1} \mathbf{p}_k)^{[\frac{1}{\alpha-1}]}, \tag{6.7}$$

in which $\mathcal{E}$ and $\mathcal{F}$ are the splitting tensors of tensor $\mathcal{R}$ satisfying $\mathcal{R} = \mathcal{E} - \mathcal{F}$, and $M(\mathcal{E})$ denotes the M -matrix of tensor $\mathcal{E}$.

(2) The SDNCG method [22] for solving (6.1) is presented as

$$\mathbf{x}_{k+1} = \mathbf{x}_k + g_k \mathbf{d}_k, \quad (6.8)$$

where g_k is obtained by the Wolfe-type line search in [22], and $\mathbf{d}_k = -((\alpha - 1)\hat{\mathcal{R}}_k \mathbf{x}_k^{\alpha-2})^T (\mathcal{R}_k \mathbf{x}_k^{\alpha-1} - \mathbf{p}_k) + \varrho_k \mathbf{d}_{k-1}$ with $\varrho_k = (\|((\alpha - 1)\hat{\mathcal{R}}_k \mathbf{x}_k^{\alpha-2})^T (\mathcal{R}_k \mathbf{x}_k^{\alpha-1} - \mathbf{p}_k)\|_2^2) / (\mu \|((\alpha - 1)\hat{\mathcal{R}}_k \mathbf{x}_k^{\alpha-2})^T (\mathcal{R}_k \mathbf{x}_k^{\alpha-1} - \mathbf{p}_k)\|_2^2 + \mu \|\mathbf{d}_{k-1}\|_2^2)$, in which $\mu > 1/2$.

(3) The DTNN model [23] constructed for solving (6.1) is formulated as

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{\lambda}{\alpha - 1} (\hat{\mathcal{R}}_k \mathbf{x}_k^{\alpha-2})^{-1} (\mathcal{R}_k \mathbf{x}_k^{\alpha-1} - \mathbf{p}_k), \quad (6.9)$$

with $\lambda > 0$.

(4) The MNI algorithm [25] used to solve (6.1) is specifically expressed as

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{1}{\alpha - 1} (\hat{\mathcal{R}}_k \mathbf{x}_k^{\alpha-2})^{-1} (\mathbf{e}_k + \xi \sum_{i=0}^k \mathbf{e}_i), \quad (6.10)$$

with $0 < \xi < 1$.

6.3 Theoretical Analyses

In this section, theoretical analyses are provided to illustrate that HRDND method (6.6) is convergent and robust.

6.3.1 Nonlinear Transformation

The proposed HRDND method (6.6) is transformed into a linear system with a residual error term, which is provided in the following theorem.

Theorem 6.1 *HRDND method (6.6) is transformed into a linear system with a residual error term as follows:*

$$\mathbf{e}_{k+1} - \left(\frac{1}{2} + \theta_1\right) \mathbf{e}_k + \mathbf{e}_{k-1} - \frac{1}{2} \mathbf{e}_{k-2} + \theta_2 \sum_{i=0}^k \mathbf{e}_i + \mathbf{O}(\delta^3) = \mathbf{0}, \quad (6.11)$$

where $\mathbf{e}_k = \mathcal{R}_k \mathbf{x}_k^{\alpha-1} - \mathbf{p}_k$; $\mathbf{O}(\delta^3) \in \mathbb{R}^b$ composed of $\mathbf{O}(\delta^3)$ is a vector.

Proof The HRDND method (6.6) is rewritten as

$$\begin{aligned} \chi_{k+1} - \frac{3}{2}\chi_k + \chi_{k-1} - \frac{1}{2}\chi_{k-2} = & -\frac{(\hat{\mathcal{R}}_k \chi_k^{\alpha-2})^{-1}}{\alpha-1} ((1-\theta_1)\epsilon_k \\ & + \theta_2 \sum_{i=0}^k \epsilon_i + \delta \dot{\mathcal{R}}_k \chi_k^{\alpha-1} - \delta \dot{p}_k). \end{aligned}$$

Using the Taylor expansion to describe χ_{k+1} , χ_{k-1} , and χ_{k-2} leads to

$$\begin{aligned} \chi_{k+1} &= \chi_k + \delta \dot{\chi}_k + \frac{\delta^2}{2} \ddot{\chi}_k + \mathcal{O}(\delta^3), \\ \chi_{k-1} &= \chi_k - \delta \dot{\chi}_k + \frac{\delta^2}{2} \ddot{\chi}_k + \mathcal{O}(\delta^3), \\ \chi_{k-2} &= \chi_k - 2\delta \dot{\chi}_k + \frac{4\delta^2}{2} \ddot{\chi}_k + \mathcal{O}(\delta^3). \end{aligned}$$

Based on the above four equations, we have

$$\begin{aligned} \delta \dot{\chi}_k + \mathcal{O}(\delta^3) = & -\frac{1}{\alpha-1} (\hat{\mathcal{R}}_k \chi_k^{\alpha-2})^{-1} ((1-\theta_1)\epsilon_k + \theta_2 \sum_{i=0}^k \epsilon_i \\ & + \delta \dot{\mathcal{R}}_k \chi_k^{\alpha-1} - \delta \dot{p}_k), \end{aligned}$$

which is transformed into

$$\begin{aligned} (\alpha-1)(\hat{\mathcal{R}}_k \chi_k^{\alpha-2}) \dot{\chi}_k + \dot{\mathcal{R}}_k \chi_k^{\alpha-1} - \dot{p}_k = & -\frac{1}{\delta} ((1-\theta_1)\epsilon_k \\ & + \theta_2 \sum_{i=0}^k \epsilon_i + \mathcal{O}(\delta^3)). \end{aligned} \tag{6.12}$$

According to (6.4), we have

$$\dot{\epsilon}_k = (\alpha-1)(\hat{\mathcal{R}}_k \chi_k^{\alpha-2}) \dot{\chi}_k + \dot{\mathcal{R}}_k \chi_k^{\alpha-1} - \dot{p}_k. \tag{6.13}$$

Combining equations (6.12) and (6.13), we obtain

$$\dot{\epsilon}_k = -\frac{1}{\delta} ((1-\theta_1)\epsilon_k + \theta_2 \sum_{i=0}^k \epsilon_i + \mathcal{O}(\delta^3)).$$

Substituting $\dot{\mathbf{e}}_k$ with Taylor-type three-step forward difference formula leads to

$$\begin{aligned}\dot{\mathbf{e}}_k &= \frac{2\mathbf{e}_{k+1} - 3\mathbf{e}_k + 2\mathbf{e}_{k-1} - \mathbf{e}_{k-2}}{2\delta} \\ &= -\frac{1}{\delta}((1 - \theta_1)\mathbf{e}_k + \theta_2 \sum_{i=0}^k \mathbf{e}_i + \mathcal{O}(\delta^3)),\end{aligned}$$

which is converted into

$$\mathbf{e}_{k+1} - \left(\frac{1}{2} + \theta_1\right)\mathbf{e}_k + \mathbf{e}_{k-1} - \frac{1}{2}\mathbf{e}_{k-2} + \theta_2 \sum_{i=0}^k \mathbf{e}_i + \mathcal{O}(\delta^3) = \mathbf{0}. \quad (6.14)$$

It is evident that (6.14) has the property of linearity, which illustrates that the principle of superposition is applicable. Thus, the proof is completed.

6.3.2 Convergence Analyses

For the DFTMTE with $\mathcal{M}$ -tensor (6.1), if the order of the coefficient $\mathcal{M}$ -tensor $\mathcal{R}$ is odd, then it can be concluded that χ^* and $-\chi^*$ are all its solutions, which illustrates that the solution to the DFTMTE with $\mathcal{M}$ -tensor (6.1) may not be unique. Therefore, the following set is given to guarantee that $\hat{\mathcal{R}}_k \chi_k^{\alpha-2}$ is nonsingular, which ensures that the DFTMTE with $\mathcal{M}$ -tensor (6.1) has a unique solution:

$$\Omega(\chi^*, \delta) = \{\chi \in \mathbb{R}^\beta \mid \|\chi - \chi^*\|_2 \leq \delta\}.$$

Then, the convergence of HRDND method (6.6) can be further proved as follows.

Theorem 6.2 *The steady-state error $\lim_{k \rightarrow \infty} \|\mathbf{e}_k\|_2$ of HRDND method (6.6) for solving the DFTMTE with $\mathcal{M}$ -tensor (6.1) in a noise-free environment is $\mathcal{O}(\delta^3)$ for any $\chi(0)$ satisfying $\chi(0) \in \Omega(\chi^*, \delta)$.*

Proof The m th subsystems of equation (6.11) at $t = (k+1)\delta$ and $t = k\delta$ are derived as follows:

$$\mathbf{e}_{k+1}^m - \left(\frac{1}{2} + \theta_1\right)\mathbf{e}_k^m + \mathbf{e}_{k-1}^m - \frac{1}{2}\mathbf{e}_{k-2}^m + \theta_2 \sum_{i=0}^k \mathbf{e}_i^m + \mathcal{O}(\delta^3) = 0 \quad (6.15)$$

and

$$\mathbf{e}_k^m - \left(\frac{1}{2} + \theta_1\right)\mathbf{e}_{k-1}^m + \mathbf{e}_{k-2}^m - \frac{1}{2}\mathbf{e}_{k-3}^m + \theta_2 \sum_{i=0}^{k-1} \mathbf{e}_i^m + \mathcal{O}(\delta^3) = 0, \text{ respectively.} \quad (6.16)$$

Then, the result of subtracting (6.16) from (6.15) is acquired as

$$\begin{aligned} \boldsymbol{\varepsilon}_{k+1}^m - \left(\frac{3}{2} + \theta_1 - \theta_2\right) \boldsymbol{\varepsilon}_k^m + \left(\frac{3}{2} + \theta_1\right) \boldsymbol{\varepsilon}_{k-1}^m - \frac{3}{2} \boldsymbol{\varepsilon}_{k-2}^m \\ + \frac{1}{2} \boldsymbol{\varepsilon}_{k-3}^m + O(\delta^3) = 0. \end{aligned} \quad (6.17)$$

Define $\mathbf{q}_{k+1}^m = [\boldsymbol{\varepsilon}_{k+1}^m; \boldsymbol{\varepsilon}_k^m; \boldsymbol{\varepsilon}_{k-1}^m; \boldsymbol{\varepsilon}_{k-2}^m]$. Then, (6.17) is described as

$$\mathbf{q}_{k+1}^m = W \mathbf{q}_k^m + O(\delta^3), \quad (6.18)$$

where

$$W = \begin{bmatrix} \frac{3}{2} + \theta_1 - \theta_2 & -(\frac{3}{2} + \theta_1) & \frac{3}{2} - \frac{1}{2} \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

According to (6.18), it is generated that

$$\begin{aligned} \mathbf{q}_{k+1}^m &= W \mathbf{q}_k^m + O(\delta^3) \\ &= W(W \mathbf{q}_{k-1}^m + O(\delta^3)) + O(\delta^3) \\ &= W^2 \mathbf{q}_{k-1}^m + O(\delta^3) \\ &= W^3 \mathbf{q}_{k-2}^m + O(\delta^3) \\ &\vdots \\ &= W^k \mathbf{q}_1^m + O(\delta^3). \end{aligned}$$

Through calculations, we obtain the four eigenvalues of matrix W , whose real parts all fall into the range $[-1, 1]$. In such a state, it is concluded that $\lim_{k \rightarrow \infty} W^k = 0$ according to the spectral radius theorem, which leads to $\mathbf{q}_{k+1}^m = O(\delta^3)$. Then, the following conclusion is further deduced as

$$\lim_{k \rightarrow \infty} \|\mathbf{q}_{k+1}^m\|_2 = O(\delta^3),$$

which is combined with the definition $\mathbf{q}_{k+1}^m = [\boldsymbol{\varepsilon}_{k+1}^m; \boldsymbol{\varepsilon}_k^m; \boldsymbol{\varepsilon}_{k-1}^m; \boldsymbol{\varepsilon}_{k-2}^m]$ to conclude $\lim_{k \rightarrow \infty} \|\boldsymbol{\varepsilon}_k\|_2 = O(\delta^3)$. Thus, the proof is completed.

6.3.3 Robustness Analyses

HRDND method (6.6) involves a sum term, which is helpful for immunity to noises. Therefore, the HRDND method has great robust performance, which is theoretically proved by the following theorem.

Theorem 6.3 For the DFTMTE with $\mathcal{M}$ -tensor (6.1) solved by HRDND method (6.6), its steady-state error $\|\mathbf{e}_k\|_2$ is bounded by $\|\mathbf{a}\delta/\theta_2\|_2 + 2\beta \sup_{1 \leq n \leq k, 1 \leq m \leq \beta} |\boldsymbol{\psi}_n^m|/(1 - \|W\|_2) + O(\delta^3)$ with $k \rightarrow \infty$ in an environment polluted by noise $\mathbf{f}_k = \mathbf{a}k\delta + \mathbf{b} + \boldsymbol{\psi}$ with $\boldsymbol{\chi}(0) \in \Omega(\boldsymbol{\chi}^*, \delta)$, in which $\mathbf{a}k\delta + \mathbf{b}$ denotes linear noise and $\boldsymbol{\psi}$ represents other noises to the system, respectively.

Proof Considering the linearity property of (6.14), the steady-state error $\|\mathbf{e}_k\|_2$ with $k \rightarrow \infty$ in a noisy environment can be divided into $O(\delta^3)$ caused by HRDND method (6.6) itself, $\|\mathbf{a}\delta/\theta_2\|_2$ generated by a linear noise, and $2\beta \sup_{1 \leq n \leq k, 1 \leq m \leq \beta} |\boldsymbol{\psi}_n^m|/(1 - \|W\|_2)$ resulting from other noises. Hence, we discuss from the following two aspects:

- (1) **Linear noise:** Equation (6.14) combined with the linear noise $\mathbf{f}_k = \mathbf{a}k\delta + \mathbf{b}$ is redesigned as

$$\begin{aligned} \mathbf{e}_{k+1} = & \left(\frac{1}{2} + \theta_1\right) \mathbf{e}_k - \mathbf{e}_{k-1} + \frac{1}{2} \mathbf{e}_{k-2} - \theta_2 \sum_{i=0}^k \mathbf{e}_i \\ & + \mathbf{a}k\delta + \mathbf{b}, \end{aligned} \quad (6.19)$$

whose m -subsystem is

$$\begin{aligned} \mathbf{e}_{k+1}^m = & \left(\frac{1}{2} + \theta_1\right) \mathbf{e}_k^m - \mathbf{e}_{k-1}^m + \frac{1}{2} \mathbf{e}_{k-2}^m - \theta_2 \sum_{i=0}^k \mathbf{e}_i^m \\ & + \mathbf{a}^m k\delta + \mathbf{b}^m. \end{aligned} \quad (6.20)$$

Then, Z-transformation is applied to elicit

$$\begin{aligned} & z\mathbf{e}^m(z) - z\mathbf{e}^m(0) \\ = & \left(\frac{1}{2} + \theta_1\right) \mathbf{e}^m(z) - z^{-1}(\mathbf{e}^m(z) + z\mathbf{e}^m(-1)) \\ & + \frac{1}{2} z^{-2}(\mathbf{e}^m(z) + z^2\mathbf{e}^m(-2) + z\mathbf{e}^m(-1)) - \theta_2 \frac{z\mathbf{e}^m(z)}{z-1} \\ & + \frac{z\mathbf{a}^m\delta}{(z-1)^2} + \frac{z\mathbf{b}^m}{z-1}, \end{aligned}$$

which is reformulated as

$$\begin{aligned}
\mathbf{e}^m(z) &= (2z^3(z-1)^2\mathbf{e}^m(0) + (1-2z)z(z-1)^2\mathbf{e}^m(-1) \\
&\quad + z^2(z-1)^2\mathbf{e}^m(-2) + 2z^3\mathbf{a}^m\delta + 2z^3(z-1)\mathbf{b}^m) \\
&\quad / ((z-1)(2z^4 - (3+2\theta_1-2\theta_2)z^3 + (3+2\theta_1)z^2 \\
&\quad - 3z + 1)).
\end{aligned}$$

Based on the final value theorem, we have

$$\begin{aligned}
\lim_{k \rightarrow \infty} \mathbf{e}_k^m &= \lim_{z \rightarrow 1} (z-1)\mathbf{e}^m(z) \\
&= \frac{2\mathbf{a}^m\delta}{2-3-2\theta_1+2\theta_2+3+2\theta_1-3+1} \\
&= \frac{\mathbf{a}^m\delta}{\theta_2}.
\end{aligned}$$

Combining the above conclusion and $O(\delta^3)$ existing in HRDND method (6.6), we can obtain that the steady-state error is $\|\mathbf{a}\delta/\theta_2\|_2 + O(\delta^3)$ with linear noise $\mathbf{f}_k = \mathbf{a}k\delta + \mathbf{b}$. Notice that the constant noise is a subcase of linear noise $\mathbf{f}_k = \mathbf{a}k\delta + \mathbf{b}$ with $\mathbf{a} = \mathbf{0}$. As a result, the steady-state error of HRDND method (6.6) for solving the DFTMTE with $\mathcal{M}$ -tensor (6.1) polluted by constant noise is $O(\delta^3)$.

- (2) **Other noise:** Similarly to (6.20), the m -subsystem of (6.14) under other noise $\mathbf{f}_k = \boldsymbol{\psi}$ is represented as

$$\mathbf{e}_{k+1}^m = \left(\frac{1}{2} + \theta_1\right) \mathbf{e}_k^m - \mathbf{e}_{k-1}^m + \frac{1}{2} \mathbf{e}_{k-2}^m - \theta_2 \sum_{i=0}^k \mathbf{e}_i^m + \boldsymbol{\psi}_k^m. \quad (6.21)$$

Then, the following equation is further deduced:

$$\mathbf{e}_k^m = \left(\frac{1}{2} + \theta_1\right) \mathbf{e}_{k-1}^m - \mathbf{e}_{k-2}^m + \frac{1}{2} \mathbf{e}_{k-3}^m - \theta_2 \sum_{i=0}^{k-1} \mathbf{e}_i^m + \boldsymbol{\psi}_{k-1}^m. \quad (6.22)$$

Subtracting (6.22) from (6.21) leads to

$$\begin{aligned}
\mathbf{e}_{k+1}^m &= \left(\frac{3}{2} + \theta_1 - \theta_2\right) \mathbf{e}_k^m - \left(\frac{3}{2} + \theta_1\right) \mathbf{e}_{k-1}^m + \frac{3}{2} \mathbf{e}_{k-2}^m \\
&\quad - \frac{1}{2} \mathbf{e}_{k-3}^m + \boldsymbol{\psi}_k^m - \boldsymbol{\psi}_{k-1}^m.
\end{aligned} \quad (6.23)$$

Define $\mathbf{g}_k^m = [\boldsymbol{\psi}_k^m - \boldsymbol{\psi}_{k-1}^m; 0]$. Then, relate it to Theorem 6.2 and we get

$$\mathbf{q}_{k+1}^m = W\mathbf{q}_k^m + \mathbf{g}_k^m.$$

From which, we have

$$\begin{aligned}
 \mathbf{q}_{k+1}^m &= W\mathbf{q}_k^m + \mathbf{g}_k^m \\
 &= W(W\mathbf{q}_{k-1}^m + \mathbf{g}_{k-1}^m) + \mathbf{g}_k^m \\
 &= W^2\mathbf{q}_{k-1}^m + W\mathbf{g}_{k-1}^m + \mathbf{g}_k^m \\
 &\quad \vdots \\
 &= W^k\mathbf{q}_1^m + \sum_{n=0}^{k-1} W^n\mathbf{g}_{k-n}^m.
 \end{aligned}$$

According to Theorem 6.2, we know $\lim_{k \rightarrow \infty} W^k = 0$. Therefore, the above formula is transformed into

$$\lim_{k \rightarrow \infty} \mathbf{q}_{k+1}^m = \sum_{n=0}^{k-1} W^n \mathbf{g}_{k-n}^m. \quad (6.24)$$

Based on (6.24), we get

$$\begin{aligned}
 \lim_{k \rightarrow \infty} \|\mathbf{q}_{k+1}^m\|_2 &= \left\| \sum_{n=0}^{k-1} W^n \mathbf{g}_{k-n}^m \right\|_2 \\
 &\leq \sum_{n=0}^{k-1} \|W^n\|_2 \max_{1 \leq n \leq k} \|\mathbf{g}_n^m\|_2 \\
 &= (1 - \|W\|_2^k) / (1 - \|W\|_2) \max_{1 \leq n \leq k} \|\mathbf{g}_n^m\|_2 \\
 &< \max_{1 \leq n \leq k} \|\mathbf{g}_n^m\|_2 / (1 - \|W\|_2) \\
 &< 2 \max_{1 \leq n \leq k} |\boldsymbol{\psi}_n^m| / (1 - \|W\|_2).
 \end{aligned}$$

Lastly, in view of the term $O(\delta^3)$ that is the residual error of HRDND method (6.6) itself and the definition of $\boldsymbol{\psi}_k^m$, we have

$$\lim_{k \rightarrow \infty} \|\boldsymbol{\epsilon}_k\|_2 < 2\beta \sup_{1 \leq n \leq k, 1 \leq m \leq \beta} |\boldsymbol{\psi}_n^m| / (1 - \|W\|_2) + O(\delta^3).$$

To sum up, under the condition polluted by noise $\mathbf{f}_k = \mathbf{a}k\delta + \mathbf{b} + \boldsymbol{\psi}$ with $\boldsymbol{\chi}(0) \in \Omega(\boldsymbol{\chi}^*, \delta)$, the steady-state error of HRDND method (6.6) for solving problem (6.1) with $\mathcal{M}$ -tensor is bounded by $\|\mathbf{a}\delta/\theta_2\|_2 + 2\beta \sup_{1 \leq n \leq k, 1 \leq m \leq \beta} |\boldsymbol{\psi}_n^m| / (1 - \|W\|_2) + O(\delta^3)$. Thus, the proof is completed.

6.4 Simulation Verifications

To verify the superiority of the given HRDND method (6.6) for solving the DFTMTE with $\mathcal{M}$ -tensor (6.1), three time-varying numerical examples are designed and simulated by the proposed HRDND method (6.6), TS method (6.7), SDNCG model (6.8), DTNN model (6.9), and MNI algorithm (6.10) in this section.

6.4.1 Example 1

In this example, the initial objective vector is set to $\mathbf{x}_0 = \text{rand}(10, 1) - 0.5$ and the simulation time is set to 3 s. Besides, a time-varying tensor with 2 orders and 10 dimensions is considered, which is described as follows with $t = k\delta$:

$$\begin{cases} \mathcal{R}_k(ij) = i + \sin(k\delta), i = j, \\ \mathcal{R}_k(ij) = \frac{\sin(k\delta)}{i-j}, i < j, \\ \mathcal{R}_k(ij) = \frac{\cos(k\delta)}{i-j}, i > j. \end{cases}$$

The positive vector $\mathbf{p}_k$ is described as

$$\mathbf{p}_k(i) = \sin(k\delta), i = 1, 2, \dots, 10.$$

Convergence performances with different sampling gaps synthesized by HRDND method (6.6), TS method (6.7), SDNCG method (6.8), DTNN model (6.9), and MNI algorithm (6.10) for solving the DFTMTE with $\mathcal{M}$ -tensor with different noises polluted in this example are presented in Fig. 6.1. From Fig. 6.1a, it can be observed that the steady-state error of HRDND method (6.6) is around the order of 10^{-9} , whereas the steady-state errors of TS method (6.7), SDNCG method (6.8), DTNN model (6.9), and MNI algorithm (6.10) are around the order of 10^0 , 10^{-3} , 10^{-3} , and 10^{-4} , respectively, which are 10^9 times, 10^6 times, 10^6 times, and 10^5 times larger than that of HRDND method (6.6), respectively. In addition, from Fig. 6.1a–f, it is obvious that HRDND method (6.6) has the best performance. Notice that, from Fig. 6.1a and d, we can find that the steady-state error of HRDND method (6.6) changes from 10^{-9} to 10^{-12} as δ changes from 0.001 to 0.0001 s, which gives the proof of Theorem 6.2 on a numerical example.

6.4.2 Example 2

In this example, the time-varying $\mathcal{M}$ -tensor $\mathcal{R}_k$ with 3 orders and 3 dimensions is given, whose non-diagonal elements are all 0 and diagonal elements are defined as follows:

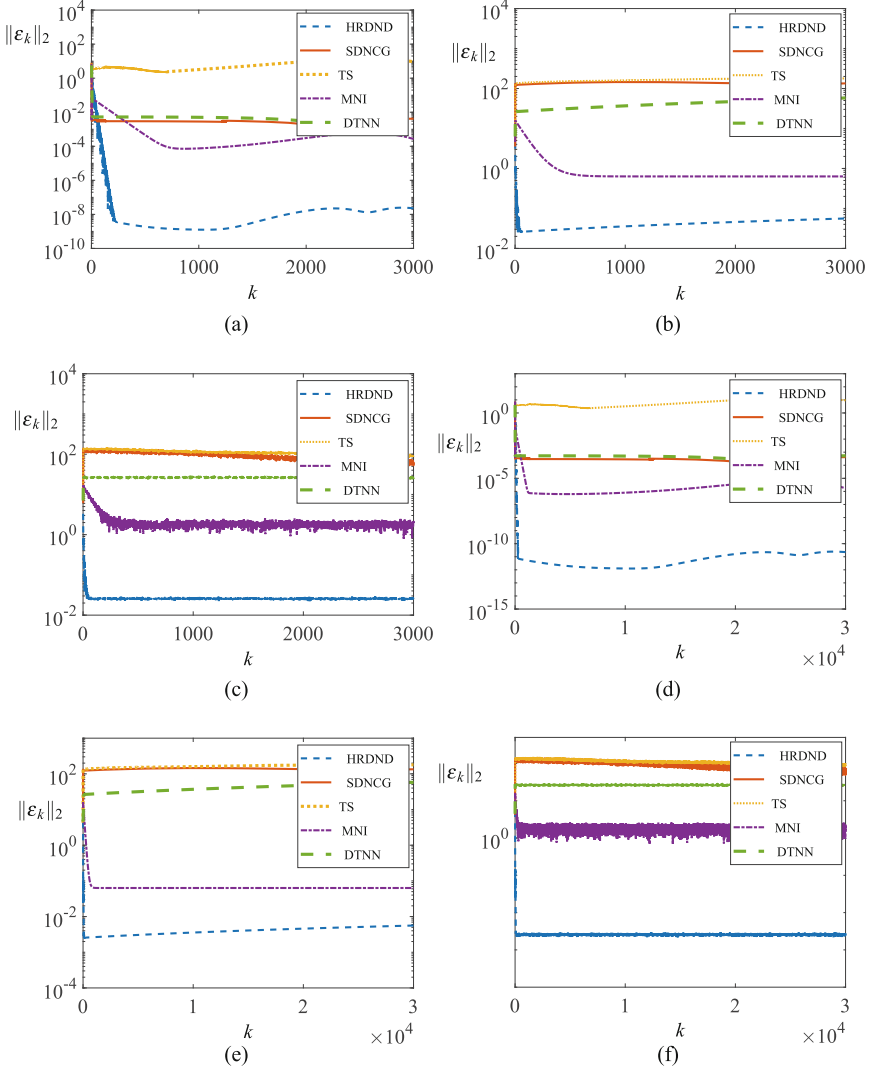


Fig. 6.1 Simulation results of HRDND method (6.6), TS method (6.7), SDNCG model (6.8), DTNN model (6.9), and MNI algorithm (6.10) for solving the DFTMTE with $\mathcal{M}$ -tensor in **Example 1** in different environments polluted by noises with different iterations. (a) No noise, $k = 3000$; (b) linear noise $f_k = 2k\delta + 5$, $k = 3000$; (c) random noise $f_k \in [4, 6]$, $k = 3000$; (d) no noise, $k = 30,000$; (e) linear noise $f_k = 2k\delta + 5$, $k = 30,000$; and (f) random noise $f_k \in [4, 6]$, $k = 30,000$

$$\begin{cases} \mathcal{R}_k(111) = 1 + \cos^2(k\delta), \\ \mathcal{R}_k(222) = 1 + \sin^2(k\delta), \\ \mathcal{R}_k(333) = 1 + \cos^2(2k\delta). \end{cases}$$

Then, the time-varying positive vector $\mathbf{p}_k$ is defined as

$$\begin{cases} \mathbf{p}_k(1) = 2 + \sin(k\delta), \\ \mathbf{p}_k(2) = 2 + \cos(k\delta), \\ \mathbf{p}_k(3) = 2 + \sin(2k\delta). \end{cases}$$

Besides, the initial state of $\mathbf{x}$ is given as $\mathbf{x}_0 = 1 + \text{rand}(3, 1)$, and the whole simulation process takes 10 s.

In order to visually display the results of HRDND method (6.6), TS method (6.7), SDNCG model (6.8), DTNN model (6.9), and MNI algorithm (6.10) for solving the DFTMTE with $\mathcal{M}$ -tensor in this example, their convergence performances are shown in Fig. 6.2. As shown in Fig. 6.2a, we can find that the steady-state error of the proposed HRDND method (6.6) remains stable around the order of 10^{-11} . In contrast, TS method (6.7), SDNCG model (6.8), DTNN model (6.9), and MNI algorithm (6.10) converge to the theoretical solution with steady-state errors being around the order of 10^0 , 10^{-4} , 10^{-4} , and 10^{-5} , respectively, which are much larger than that of HRDND method (6.6). From Fig. 6.2a and d, it is found that the steady-state error of HRDND method (6.6) changes from 10^{-11} to 10^{-15} when δ changes from 0.01 to 0.001 s. Furthermore, we can find that steady-state errors of the other four methods are much larger than that of the proposed HRDND method (6.6) both with and without noise pollution from Fig. 6.2a–f, which verifies the superiority of the given HRDND method (6.6).

6.4.3 Application

In this part, a Bellman equation is discretized as a TMTE, whose coefficient tensor with 3 orders and β dimensions is given as follows when $t = k\delta$:

$$\begin{cases} \mathcal{R}_k(1, 1, 1) = 1, \\ \mathcal{R}_k(m, m, m) = (\frac{f}{s})^2 + \frac{(k\delta)^2}{s}, \\ \mathcal{R}_k(m, m, m-1) = -0.25((\frac{f}{s})^2) + 0.5\frac{(k\delta)^2}{s}, \\ \mathcal{R}_k(m, m-1, m) = \mathcal{R}_k(m, m, m-1), 2 \leq m \leq \beta-2, \\ \mathcal{R}_k(m, m, m+1) = -0.25((\frac{f}{s})^2) + 0.5\frac{(k\delta)^2}{s}, \\ \mathcal{R}_k(m, m+1, m) = \mathcal{R}_k(m, m, m+1), \\ \mathcal{R}_k(\beta, \beta, \beta) = 1, \end{cases}$$

where $f = 0.2$ and $s = \frac{2}{\beta}$. The right-hand-side positive vector is defined as

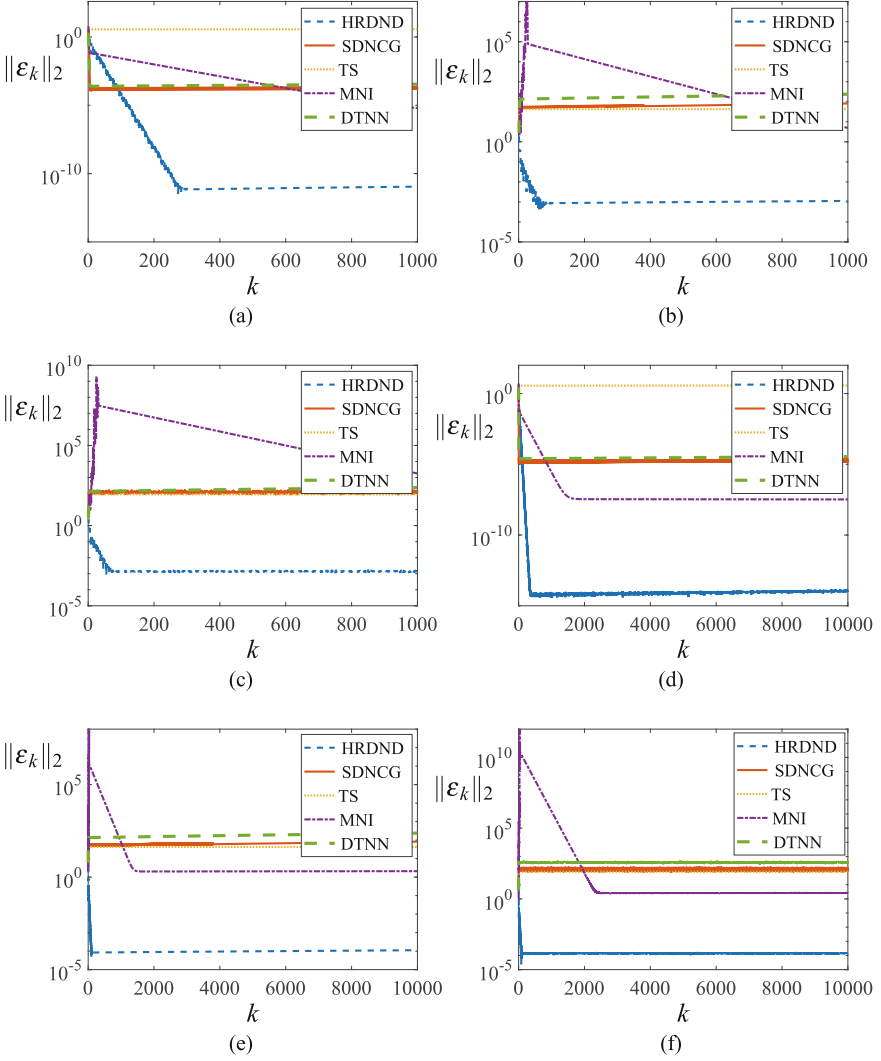


Fig. 6.2 Simulation results of HRDND method (6.6), TS method (6.7), SDNCG model (6.8), DTNN model (6.9), and MNI algorithm (6.10) for solving the DFTMTE with $\mathcal{M}$ -tensor in **Example 2** in different environments polluted by noises with different iterations. (a) no noise, $k = 1000$; (b) linear noise $f_k = 10k\delta + 3$, $k = 1000$; (c) random noise $f_k \in [4, 6]$, $k = 1000$; (d) no noise, $k = 10,000$; (e) linear noise $f_k = 10k\delta + 3$, $k = 10,000$; and (f) random noise $f_k \in [4, 6]$, $k = 10,000$

$$\begin{cases} p_k(1) = 1, \\ p_k(i) = \frac{w^2}{2v}, \quad 1 < i < \beta, \\ p_k(\beta) = 1, \end{cases}$$

in which $w = 2$ and $v = 1$.

In this application, β is set to 4, and the initial state of χ is given as $\chi_0 = \text{ones}(4, 1)$. Besides, the whole simulation process takes 0.01 s. Then, convergence results with different sampling gaps of HRDND method (6.6), TS method (6.7), SDNCG method (6.8), DTNN model (6.9), and MNI algorithm (6.10) for solving the DFTMTE with $\mathcal{M}$ -tensor polluted by different noises are summarized in Fig. 6.3. From Fig. 6.3a, it is obtained that the steady-state error of HRDND method (6.6) is stable at 10^{-5} , whereas that of SDNCG method (6.8) as well as DTNN model (6.9) is stable at 10^{-4} , and the steady-state error of TS method (6.7) converges to 10. Notice that MNI algorithm (6.10) is not effective in solving this application. In addition, from Fig. 6.3a and d, we find that the steady-state error of HRDND method (6.6) changes from 10^{-4} to 10^{-5} when δ varies from 1×10^{-5} to 1×10^{-6} s, and steady-state errors of SDNCG method (6.8) and DTNN model (6.9) are close to that of HRDND method (6.6). However, it is obvious that the steady-state error of HRDND method (6.6) is much smaller than that of other methods when this application is polluted by noises from Fig. 6.3b–c and e–f, which illustrates the great robustness of HRDND method (6.6).

6.5 Conclusion

In this chapter, in order to accommodate facts that some problems are discrete in nature and noises are inevitable in solving process, the HRDND method has been designed from a perspective of a directly discrete way with the aid of the Taylor-type difference formula, which solves multi-linear tensor equations with high accuracy. In addition, the corresponding convergence and robustness have been proved by theoretical analyses and practically demonstrated by numerical examples. Then, the proposed HRDND method has been compared with the TS method, SDNCG model, DTNN model, and MNI algorithm, which illustrates that the HRDND method is superior to the other four methods for solving the DFTMTE with $\mathcal{M}$ -tensor.

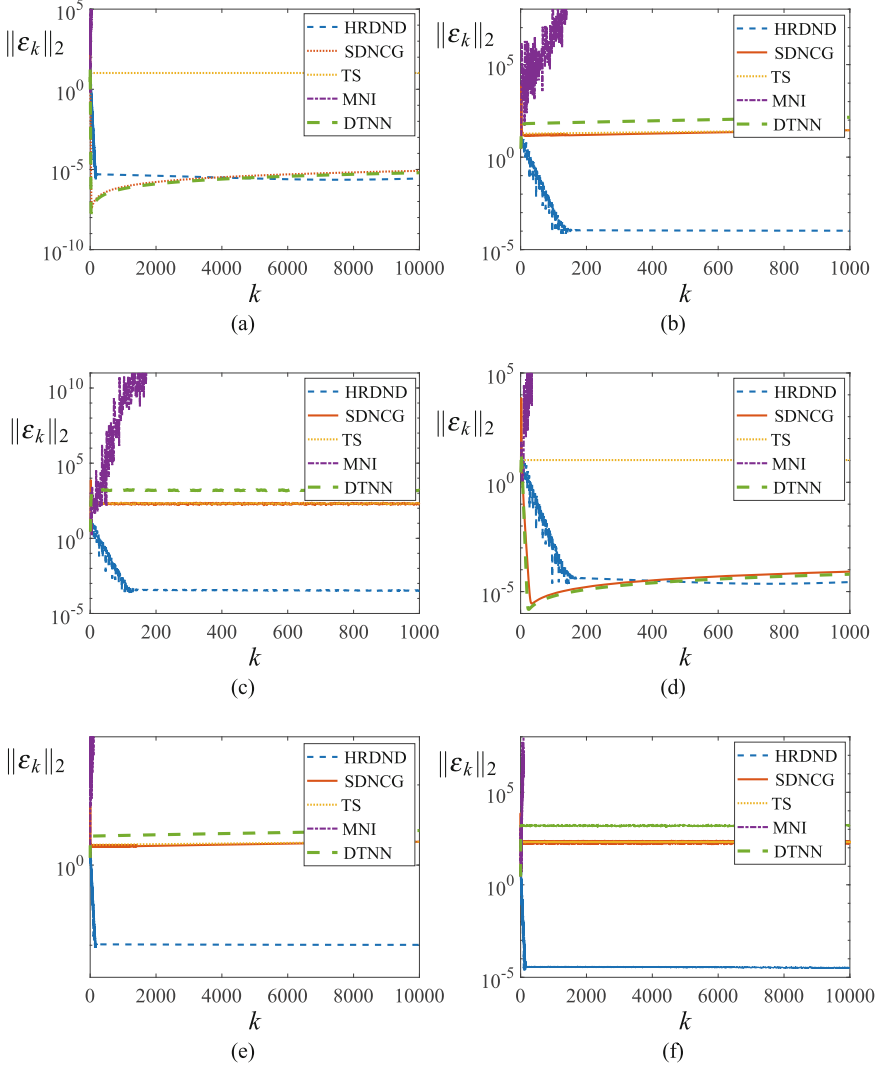


Fig. 6.3 Simulation results of HRDND method (6.6), TS method (6.7), SDNCG model (6.8), DTNN model (6.9), and MNI algorithm (6.10) for solving the DFTMTE with $\mathcal{M}$ -tensor in **Application** in different environments polluted by noises with different iterations. (a) No noise, $k = 1000$; (b) linear noise $f_k = 100k\delta + 2$, $k = 1000$; (c) random noise $f_k \in [9, 11]$, $k = 1000$; (d) no noise, $k = 10000$; (e) linear noise $f_k = 100k\delta + 2$, $k = 10000$; and (f) random noise $f_k \in [9, 11]$, $k = 1000$

References

1. Wang X, Yang LT, Song L, Wang H, Ren L, Deen MJ (2021) A tensor-based multi-tributes visual feature recognition method for industrial intelligence. *IEEE Trans Ind Inform* 17(3):2231–2241
2. Ma Z, Yang LT, Zhang Q (2021) Support multimode tensor machine for multiple classification on industrial big data. *IEEE Trans Ind Inform* 17(5):3382–3390
3. Kong L, Liu X, Sheng H, Zeng P, Chen G (2020) Federated tensor mining for secure industrial internet of things. *IEEE Trans Ind Inform* 16(3):2144–2153
4. Zhao Y, Yang LT, Sun J (2019) Privacy-preserving tensor-based multiple clusterings on cloud for industrial IoT. *IEEE Trans Ind Inform* 15(4):2372–2381
5. Ren L, Jia Z, Wang X, Dong J, Wang W (2022) A T^2 -tensor-aided multi-scale transformer for remaining useful life prediction in IIoT. *IEEE Trans Ind Inform* 18(11):8108–8118
6. Zheng Y, Sun Y, Muhammad K, de Albuquerque VHC (2021) Weighted LIC-based structure tensor with application to image content perception and processing. *IEEE Trans Ind Inform* 17(3):2250–2260
7. Ahmed J, Gao B, Woo WL (2021) Sparse low-rank tensor decomposition for metal defect detection using thermographic imaging diagnostics. *IEEE Trans Ind Inform* 17(3):1810–1820
8. Madbhavi R, Natarajan B, Srinivasan B (2021) Enhanced tensor completion based approaches for state estimation in distribution systems. *IEEE Trans Ind Inform* 17(9):5938–5947
9. Gai K, Qiu M (2018) Blend arithmetic operations on tensor-based fully homomorphic encryption over real numbers. *IEEE Trans Ind Inform* 14(8):3590–3598
10. Shi Q, Cheung Y, Zhao Q, Lu H (2019) Feature extraction for incomplete data via low-rank tensor decomposition with feature regularization. *IEEE Trans Neural Netw Learn Syst* 30(6):1803–1817
11. Wang X, Yang LT, Wang Y, Ren L, Deen MJ (2021) ADTT: a highly efficient distributed tensor-train decomposition method for IIoT big data. *IEEE Trans Ind Inform* 17(3):1573–1582
12. Le TN, Giap DB, Wang J-W, Wang C-C (2021) Tensor-compensated color face recognition. *IEEE Trans Inf Foren Secur* 16:3339–3354
13. Schnabel RB, Frank PD (1984) Tensor methods for nonlinear equations. *SIAM J Numer Anal* 21(5):815–843
14. Bader B (2005) Tensor-Krylov methods for solving large-scale systems of nonlinear equations. *SIAM J Numer Anal* 43(3):1321–1347
15. Xie Z, Jin X, Wei Y (2018) Tensor methods for solving symmetric $\mathcal{M}$ -tensor systems. *J Sci Comput* 74:412–425
16. Ding W, Wei Y (2016) Solving multi-linear systems with $\mathcal{M}$ -tensors. *J Sci Comput* 68(2):689–715
17. He H, Ling C, Qi L, Zhou G (2018) A globally and quadratically convergent algorithm for solving multilinear systems with $\mathcal{M}$ -tensors. *J Sci Comput* 76(3):1718–1741
18. Li D, Xie S, Xu H (2017) Splitting methods for tensor equations. *Numer Linear Algebra Appl* 24(5):e2102
19. Liu D, Li W, Vong S (2018) The tensor splitting with application to solve multi-linear systems. *J Comput Appl Math* 330:75–94
20. Lv C, Ma C (2018) A Levenberg-Marquardt method for solving semi-symmetric tensor equations. *J Comput Appl Math* 332:13–25
21. Liang M, Zheng B, Zheng Y, Zhao R (2021) A two-step accelerated Levenberg-Marquardt method for solving multilinear systems in tensor-train format. *J Comput Appl Math* 382:1–19
22. Liu L, Du S, Chen Y (2020) A sufficient descent nonlinear conjugate gradient method for solving $\mathcal{M}$ -tensor equations. *J Comput Appl Math* 371:1–11
23. Wang X, Che M, Wei Y (2019) Neural networks based approach solving multi-linear systems with $\mathcal{M}$ -tensors. *Neurocomputing* 351:33–42
24. Wang X, Che M, Wei Y (2020) Neural network approach for solving nonsingular multi-linear tensor systems. *J Comput Appl Math* 368:1–20

25. Huang H, Fu D, Zhang J, Xiao X, Wang G, Liao S (2020) Modified Newton integration neural algorithm for solving the multi-linear $\mathcal{H}$ -tensor equation. *Appl Soft Comput J* 96:1–12
26. Qi Y, Jin L, Li H, Li Y, Liu M (2020) Discrete computational neural dynamics models for solving time-dependent Sylvester equations with applications to robotics and MIMO systems. *IEEE Trans Ind Inform* 16(10):6231–6241
27. Jin L, Yan J, Du X, Xiao X, Fu D (2020) RNN for solving time-variant generalized Sylvester equation with applications to robots and acoustic source localization. *IEEE Trans Ind Inform* 16(10):6359–6369
28. Shi Y, Zhao W, Li S, Li B, Sun X (2023) Novel discrete-time recurrent neural network for robot manipulator: A direct discretization technical route. *IEEE Trans Neural Netw Learn Syst* 34(6):2781–2790
29. Wei L, Jin L (2024) Robust neural differential solution for time-dependent nonlinear optimization with noises rejection. *IEEE Trans Autom Sci Eng*. In Press. <https://doi.org/10.1109/TASE.2024.3351197>
30. Liu M, Zhang X, Shang M (2022) Computational neural dynamics model for time-variant constrained nonlinear optimization applied to winner-take-all operation. *IEEE Trans Ind Inform* 18(9):5936–5948

Chapter 7

Collaborative Neural Dynamics



7.1 Introduction

Optimization problems have been a long-lasting topic and fundamental issue for various mathematical and engineering problems [1, 2] such as resource allocation [3], latent factor analysis [4], dynamic scheduling [5], etc. For instance, a subspace minimization algorithm based on the conjugate gradient is used to solve large-scale unconstrained optimizations [6], which approximates the objective function into a quadratic form and ensures global convergence. Generally, it is quite challenging to find one global optimal solution for the nonconvex optimization problem, and the challenge is escalated once the task has to be conducted in real time [7–9]. For example, the formulation of basic sparse problems for information processing including representation and recovery commonly takes the form of the nonconvex objective function, that is, L_0 -norm minimization [8]. To realize real-time tracking and optimal interaction on robots, a broad fuzzy neural network is leveraged to address the impedance optimization in an unknown environment [9]. Nowadays various optimization problems pose demands for designing robust, accurate, and efficient methods suitable for noisy and dynamic environments.

Manifold practical problems require in-depth investigations on the time-varying optimization problem that belongs to a more general form than the static one [10]. On many occasions, static optimization uses a tacit condition that the objective function and constraints are constant over a short-term time. Therefore, static optimization algorithms, for example, Newton iterative method and gradient neural network, ignore the bearing of time on solutions and amplify the solution error when handling time-varying problems [11].

The advantages of the neural-network-based method have attracted extensive attention to tap its potential for plenty of problems [12]. Hopfield and Tank are pioneers in the field of neural dynamics and introduce the “neural” computation for optimization [13]. Inspired by this, a series of neural dynamics methods has been developed for optimization problems and demonstrated as a promising alternative

[14, 15]. For example, research from Xia and Wang suggests that a projection neural method with a one-layer structure is globally stable for the nonlinear convex optimization [16]. Further, based on projection operators and penalty parameters, this method is extended to the nonsmooth optimization as long as the objective function is at least pseudoconvex [17]. Subsequently, the method given in [18] is validated by theoretical analyses and numerical simulations to allow the objective function to be nonsmooth pseudoconvex along with the quasiconvex constraints. Efforts on the time-varying optimization problem are made by many scholars from both theoretical and practical aspects. A Taylor-type neural method is chosen as suitable for online solving the time-varying quadratic programming (QP) subject to equality constraints with zero-stable proof, $O(\tau^3)$ pattern convergence error with τ denoting the sampling gap, and hardware implement ability [19]. In addition, to bypass the difficulty of handling the inequality constraint, the time-variant neural algorithm exploits the predefined-time stabilizer and Fischer-Burmeister function and is preferable to the time-critical QP problems containing applications to image fusion and robot control [20].

The use of neural dynamics methods for the generalized convex optimizations with the unimodal objective function has been fully endorsed by well-grounded research. However, pitfalls of these methods would be clearly recognized in nonconvex optimization problems with multimodal objective functions in front of noisy environments since a majority of these methods are sensitive to inescapable disturbances including the truncation error introduced by the computing device, random noise generated from inside or exogenous systems, etc. On the one hand, swarm intelligence, a stream of nature-inspired computational paradigm, is a major part of evolutionary computation belonging to artificial intelligence [21] and has popular methods such as ant colony optimization [22] and particle swarm optimization (PSO) [23]. Due to the simple collaborative rule, many attempts on the hybridization of PSO have been taken such as genetic algorithm (GA) [24], Tabu search [25], ant colony optimization [26], differential evolution (DE) [27], and other approaches [28, 29]. In this sense, there is interest in the mutual learning between the neural dynamics and PSO, which not only compensates for the disadvantages of PSO in inept local search and premature convergence, but also enhances the ability of neural dynamics in global search. On the other hand, the vast majority of works have been conducted only on the nominal design while how noises influence the solving process is left out. Detailedly, noises are unavoidable factors that perpetuate the instability of the whole system, hinder the evolution of the involved components, and degrade the solution accuracy of the implemented solver.

It is essential to consider the robustness of the nonconvex optimization algorithm against additive noises for developing reliable neural methods. Nevertheless, it remains unexplored on designing algorithms for solving time-varying nonconvex optimization problems with inherent rejection to noises. To this end, the collaborative neural dynamics proposed in this chapter integrates the neural dynamics method and PSO to avoid unbalanced exploitation and trapping in local minima and improve the exploitation tendencies. Firstly, this chapter deliberately considers the nonconvex optimization problems in a time-varying manner under the pollution

of different noises. Then, the collaborative neural dynamics is designed based on PSO and the neural dynamics method with the time-variation and robustness improvement. Finally, performance from different simulative nonconvex problems and an application to manipulability optimization based on the robot verifies the work's feasibility.

The rest of this chapter orderly constitutes the subsequent five parts. Section 7.2 begins with the basic problem introduction and follows by the designing process of the proposed collaborative neural dynamics. Subsequently, theoretical analyses formulate the Markov chain and prove the convergence of the proposed collaborative neural dynamics in Sect. 7.3. Besides, the proof on noise-tolerant properties of the neural dynamics method is given. Then, simulation verification and applications compared with other existing methods on benchmark problems are described in Sects. 7.4 and 7.5, respectively. Finally, Sect. 7.8 summarizes the work carried out in this chapter. The contributions are outlined below:

- (1) The time-varying nonlinear optimization under the nonconvex situation is taken into consideration and solved by the proposed collaborative neural dynamics.
- (2) The sound theoretical analyses are provided on how the proposed collaborative neural dynamics is bound to find an optimal solution with the probability one and how the collaborative neural dynamics handles the additive noises with the neural dynamics method.
- (3) Simulative examples and applications to nonconvex situations validate the feasibility of the proposed collaborative neural dynamics.

7.2 Problem Formulation and Model Description

This section provides terms that will be used throughout this chapter and the collaborative neural dynamics for the time-varying nonlinear optimization problem.

7.2.1 Problem Formulation

Generally, the considered time-varying optimization problem is given as

$$\begin{aligned} & \min F(\mathbf{u}(t), t) \\ & \text{s. t. } \mathbf{u}(t) \in \mathcal{E}, \end{aligned} \tag{7.1}$$

where $\mathbb{R}^n$ stands for vectors of dimension n and also n -dimensional vector space; decision variable $\mathbf{u}(t) \in \mathbb{R}^n$ is not static but changes with t , which is required to minimize the objective function $F(\cdot, \cdot) : \mathbb{R}^n \times [t_b, t_e] \rightarrow \mathbb{R}$ at each time instant; $t \in [t_b, t_e]$, where t_b and t_e represent the beginning and ending values of the time

domain, respectively; $0 \leq t_b < t_e$; $\mathcal{E}$ represents a nonempty and closed convex set of constraints and also a subset of $\mathbb{R}^n$.

Definition 1 A real-valued function $F(\cdot, \cdot) : \mathbb{R}^n \times [t_b, t_e] \rightarrow \mathbb{R}$ has a local minimum point at $\mathbf{u}^*(t_0)$ for one time instant t_0 , if $\exists \ell > 0$ such that $F(\mathbf{u}^*(t_0), t_0) \leq F(\mathbf{u}(t_0), t_0)$ for all $\mathbf{u}(t_0) \in \mathcal{E}$ within distance ℓ of $\mathbf{u}^*(t_0)$.

This means that, at the time t_0 , $\mathbf{u}^*(t_0)$ provides a value of the objective function that is lower (or at least not higher) than the values at nearby points. Note that the majority of neural dynamics methods specialize in local optimization situation. For example, the zeroing neural dynamics starts with an arbitrary initialization, search for the optimal solution by minimizing the energy function, and ends until that certain conditions are satisfied [19].

Definition 2 A real-valued function $F(\cdot, \cdot) : \mathbb{R}^n \times [t_b, t_e] \rightarrow \mathbb{R}$ has a global minimum point at $\mathbf{u}^*(t_0)$ for one time instant t_0 , if $F(\mathbf{u}^*(t_0), t_0) \leq F(\mathbf{u}(t_0), t_0)$ for all $\mathbf{u}(t_0) \in \mathcal{E}$.

This means that, at the time t_0 , $\mathbf{u}^*(t_0)$ provides the lowest possible value of the objective function over the entire feasible region $\mathcal{E}$. Compared with the detailed and comprehensive research on the local optimization algorithm, the global optimization algorithm still needs to be further explored. Concretely, whether or not the global optimization algorithm would converge or stall in a local trap remains open to scrutiny.

7.2.2 Collaborative Neural Dynamics

The time-varying optimization problem is omnipresent in real-world applications, which results in various optimization algorithms, and yet a highly feasible and effective method is still understudied. Here a collaborative neural dynamics is given.

For clarity, a brief explanation of the symbols, superscripts, and subscripts is provided in Table 7.1. In PSO, the velocity $\mathbf{v}$, the position $\boldsymbol{\varphi}$, and objective function value $F(\boldsymbol{\varphi})$ of $\mathcal{N}$ particles at each iteration make up the evolution information. Additionally, the position of the i th particle at the k th iteration is $\boldsymbol{\varphi}^{i,k} = (\varphi_1^{i,k}, \varphi_2^{i,k}, \dots, \varphi_n^{i,k})$; the velocity of the i particle at the k th iteration lies in $\mathbf{v}^{i,k} = (v_1^{i,k}, v_2^{i,k}, \dots, v_n^{i,k})$; $\boldsymbol{\phi}_B^{i,k} = (\phi_1^{i,k}, \phi_2^{i,k}, \dots, \phi_n^{i,k})$ is the best position (abbreviated as $_B$) that the i th particle has been through during k iterations and updates by

$$\boldsymbol{\phi}_B^{i,k} = \begin{cases} \boldsymbol{\phi}_B^{i,k-1}, & F(\boldsymbol{\varphi}^{i,k}) \geq F(\boldsymbol{\phi}_B^{i,k-1}) \\ \boldsymbol{\varphi}^{i,k}, & F(\boldsymbol{\varphi}^{i,k}) < F(\boldsymbol{\phi}_B^{i,k-1}). \end{cases} \quad (7.2)$$

Table 7.1 Definition of imperative variables used in this chapter

$F(\cdot)$	Objective function
$\mathbf{u}(t) \in \mathbb{R}^n$	Decision variable with n dimension
$\boldsymbol{\varphi}^{i,k}$	Position of the i th agent at k th iteration
$\mathbf{v}^{i,k}$	Velocity of i th agent
$\boldsymbol{\phi}_B^{i,k}$	Best position (abbreviated as B) that i th agent has been through during k iterations
$\boldsymbol{\phi}_G^k$	Best position of all agents at k th iteration
$\mathcal{N}$	The number of the participated agents
$\mathfrak{w}$	Inertial weight
λ_1, λ_2	Cognitive and social learning factors
$\mathfrak{w}_1, \mathfrak{w}_2$	Numbers randomly scattered in $[0, 1]$ following uniform distribution
$\varepsilon > 0, \varsigma > 0$	Scaling factors of GNSR solution (7.5)
$\mathfrak{F}(\cdot)$	The activation function

For all particles, $\boldsymbol{\phi}_G^k = (\phi_1^k, \phi_2^k, \dots, \phi_n^k)$ is their best position at the k th iteration and determined by $\boldsymbol{\phi}_G^k = \arg \min \{F(\boldsymbol{\phi}_B^{1,k}), F(\boldsymbol{\phi}_B^{2,k}), \dots, F(\boldsymbol{\phi}_B^{\mathcal{N},k})\}$.

Then, the following equation describes how the i th particle updates itself from the k th step to $k + 1$ th step:

$$\begin{cases} \boldsymbol{\varphi}^{i,k+1} = \boldsymbol{\varphi}^{i,k} + \mathbf{v}^{i,k+1} \\ \mathbf{v}^{i,k+1} = \mathfrak{w}\mathbf{v}^{i,k} + \lambda_1\mathfrak{w}_1(\boldsymbol{\phi}_B^{i,k} - \boldsymbol{\varphi}^{i,k}) + \lambda_2\mathfrak{w}_2(\boldsymbol{\phi}_G^k - \boldsymbol{\varphi}^{i,k}). \end{cases} \quad (7.3)$$

With further analyses, Bergh provides this equation, $1 > \mathfrak{w} > (\lambda_1 + \lambda_2)/2 - 1$, to ensure that the motion trail of the particle will converge [30]. However, this equation only guarantees that there exists a possibility for the PSO method to converge but fails to ensure that it is able to converge to a local optimum or global optimum [31].

Secondly, the gradient-based neural solution with robustness (GNSR) is given as

$$\begin{aligned} \mathcal{M}(t)\dot{\mathbf{u}}(t) = & -\varepsilon\mathcal{M}(t)\mathcal{M}^T(t)\mathfrak{F}(\boldsymbol{\rho}(t)) - \partial\boldsymbol{\rho}(t)/\partial t - \varsigma\mathfrak{F}(\boldsymbol{\rho}(t)) \\ & + \varepsilon\mathcal{M}(t)\mathcal{M}^T(t) \int_0^t \mathfrak{F}(\boldsymbol{\rho}(\tau)d\tau) \end{aligned} \quad (7.4)$$

where $\mathcal{M}(t) = \partial^2 F(\mathbf{u}(t))/\partial(\mathbf{u}(t)\mathbf{u}^T(t)) \in \mathbb{R}^{n \times n}$; $\boldsymbol{\rho}(t) = \nabla F(\mathbf{u}(t)) \in \mathbb{R}^n$. Therefore, we could get $\mathbf{u}(t)$ as

$$\begin{aligned} \dot{\mathbf{u}}(t) = \mathcal{M}^\dagger(t) \Bigg(& -\varepsilon\mathcal{M}(t)\mathcal{M}^T(t)\mathfrak{F}(\nabla F(\mathbf{u}(t))) - \partial\nabla F(\mathbf{u}(t))/\partial t - \varsigma\mathfrak{F}(\nabla F(\mathbf{u}(t))) + \\ & \varepsilon\mathcal{M}(t)\mathcal{M}^T(t) \int_0^t \mathfrak{F}(\nabla F(\mathbf{u}(t)))d\tau \Bigg). \end{aligned} \quad (7.5)$$

The specific derivation of GNSR solution (7.5) could be referred in Appendix A.

Thirdly, the collaborative neural dynamics (CND) is proposed through cooperating the GNSR (7.4) with PSO for solving nonconvex problem (7.1) with noise $\mathbf{w}(t)$ and its pseudocode is presented in Algorithm 2. Here we present three kinds of termination conditions. Condition (a) applies when we foreknow the theoretical global optimal solution ϕ_G^* , i.e., the difference between the theoretical global optimal solution ϕ_G^* and the calculated ϕ_G^k remains a tiny value, i.e., $\|\phi_G^* - \phi_G^k\|_2 \leq b$ with b being an infinitesimal value; condition (b) applies when there is a lack of prior knowledge of the theoretical global optimal solution, i.e., there is tiny or no variation of ϕ_G^k within several successive iterations; and condition (c): $k = k_{\max}$.

The collaborative neural dynamics establishes the neural community with neural agents driven by the neural dynamics method, i.e., the GNSR (7.5). In detail, for one thing, the significance of integrating PSO to construct iterative rules lies in the fact that PSO owns the swarm intelligence with the simple structure. PSO simulates the social intelligent behavior of a flock of birds. The key that the flock of birds can achieve the efficient coordination during the flight is inseparable from a single individual following simple behavioral rules and maintaining a suitable distance from other individuals, which profits from its precise perception of the local environmental change. For another, since the searching process of PSO or other swarm intelligent methods can be regarded as information processing, as an improvement, the neural dynamics method can be an effective tool to not only provide the precise local information but supervise the speed or direction for all particles perturbed by noises. It is worth noting that the GNSR (7.5) only inherits the merit of gradient descent but offsets its insufficiency of generating lagging errors when handling the time-varying optimization. Besides, the robustness improvement on GNSR (7.5) is made since it is capable of computing accurately and converging rapidly with disturbance.

7.3 Theoretical Analyses and Results

What follows are proofs of the proposed dynamic scheme on its convergence and robustness.

Definition 3 ([32]) A Markov chain or Markov process is a stochastic model describing a sequence of possible states $\{\gamma_1, \gamma_2, \gamma_3, \dots\}$ in which the probability of each state depends only on the state attained in the present:

$$\mathcal{P}(\gamma_{k+1} \mid \gamma_1, \dots, \gamma_k) = \mathcal{P}(\gamma_{k+1} \mid \gamma_k),$$

of which $\mathcal{P}(\gamma_i)$ denotes the probability of obtaining γ_i at the i th step.

Lemma 1 ([32]) Consider that a finite Markov chain as in Definition 3. γ_{nr} denotes the nonrecurrent state at the k th iteration. $\mathfrak{S}$ is the set composed of recurrent states. Thus,

Algorithm 2 Collaborative neural dynamics

Input: number of neural agents as $\mathcal{N}$ ($i = 1, 2, \dots, \mathcal{N}$), maximum iteration number $k_{\max}$, parameter $\mathfrak{w}, \mathfrak{w}_1, \mathfrak{w}_2, \lambda_1, \lambda_2, \varepsilon$, and ς

1. Initialization:

for each neural agent i from 1 to $\mathcal{N}$ **do**

Randomly initialize position $\varphi^{i,0}$

Compute objective function value $F(\varphi^{i,0})$, $\phi_B^{i,0}$, and ϕ_G^0

end for

$k \leftarrow 1$

2. Main loop:

while termination conditions not met **do**

for each neural agent i from 1 to $\mathcal{N}$ **do**

Run each $\varphi^{i,k}$ through GNSR model (7.5) with $u(t) = \varphi^{i,k}$ as

$$\begin{aligned} \frac{d\varphi^{i,k}}{dt} = & \mathcal{M}^\dagger(t) \left(-\varepsilon \mathcal{M}(t) \mathcal{M}^\top(t) \mathfrak{F}(\nabla F(\varphi^{i,k})) - \partial \nabla F(\varphi^{i,k}) / \partial t - \varsigma \mathfrak{F}(\nabla F(\varphi^{i,k})) + \right. \\ & \left. \varepsilon \mathcal{M}(t) \mathcal{M}^\top(t) \int_0^t \mathfrak{F}(\nabla F(\varphi^{i,k})) d\tau \right), \end{aligned} \quad (7.6)$$

obtain the solution $\tilde{\varphi}^{i,k}$ to (7.6), and remove solution that does not satisfy constraint Ξ

Compute the best solution that each neural agent i has been through:

$$\phi_B^{i,k} = \begin{cases} \phi_B^{i,k-1}, & F(\tilde{\varphi}^{i,k}) \geq F(\varphi^{i,k-1}) \\ \tilde{\varphi}^{i,k}, & F(\tilde{\varphi}^{i,k}) < F(\varphi^{i,k-1}) \end{cases} \quad (7.7)$$

end for

Find the best one among all agents:

$$\phi_G^k = \arg \min \{F(\phi_B^{1,k}), F(\phi_B^{2,k}), \dots, F(\phi_B^{\mathcal{N},k})\}. \quad (7.8)$$

for each neural agent i from 1 to $\mathcal{N}$ **do**

Update $\varphi^{i,k}$ according to

$$\varphi^{i,k} = \varphi^{i,k} + \mathfrak{w}(\tilde{\varphi}^{i,k} - \varphi^{i,k}) + \lambda_1 \mathfrak{w}_1(\phi_B^{i,k} - \varphi^{i,k}) + \lambda_2 \mathfrak{w}_2(\phi_G^k - \varphi^{i,k}). \quad (7.9)$$

end for

$k++$

end while

Output result ϕ_G^k and exit

$$\lim_{k \rightarrow +\infty} \mathcal{P}[\gamma_{nr} \in \mathfrak{G}] = 1. \quad (7.10)$$

The finite Markov chain starts from any nonrecurrent state that is bound to reach the recurrent state.

Theorem 1 *The proposed CND converges globally with the probability one.*

Proof Firstly, the proposed CND based on GNSR model (7.5) searches for the optimal solution in a discrete and limited problem space. For CND, its state space of

the information set is used to record the evolution information, i.e., optimal solution on each iteration, which is limited and irrelevant to time, as indicated in Definition 3. Furthermore, the whole state space χ of the CND, ensuring that it is finite, is:

$$\chi = \{(\Omega_g, \phi_B^1, \phi_B^2 \dots \phi_B^i, \dots \phi_B^{\mathcal{N}}) | \Omega_g \in \chi_g; \\ \phi_B^1, \phi_B^2 \dots \phi_B^i, \dots \phi_B^{\mathcal{N}} \in \chi_l\},$$

where Ω_g represents the global information set, and therefore, the global optimum is kept after it is found; $\phi_B^1, \phi_B^2 \dots \phi_B^i, \dots \phi_B^{\mathcal{N}}$ symbolizes the local information set; the state space for the global and local information could be denoted as set χ_g and χ_l , respectively, that are both finite.

Secondly, the state space χ is divided into the following two subsets:

$$\begin{cases} \chi_1 = \{(\Omega_g, \phi_B^1, \dots \phi_B^{\mathcal{N}}) | \Omega_g \in \chi_g^*; \phi_B^1, \dots \phi_B^{\mathcal{N}} \in \chi_l^*\} \\ \chi_2 = \{(\Omega_g, \phi_B^1, \dots \phi_B^{\mathcal{N}}) | \Omega_g \in \chi_g^*; \phi_B^1, \dots \phi_B^{\mathcal{N}} \not\in \chi_l^*\}, \end{cases}$$

of which χ_g^* is the global information set including all the global optimal solution; χ_l^* is the local information set containing global optimal solution; symbol $\not\in$ indicates that $\phi_B^1, \dots \phi_B^{\mathcal{N}}$ not all belong to χ_l^* . Clearly, $(\chi_1 \cup \chi_2) \subseteq \chi$ and $(\chi_1 \cap \chi_2) \subseteq \emptyset$ hold true. Each state γ_i in closed set χ_1 is recurrent since once a global optimum is reached, the CND retains it in the global information set Ω_g , implying that $\chi_1 \subseteq \mathfrak{G}$, as indicated in Definition 3. By the nature of the CND, any state γ_j in set χ_2 has a nonzero probability of transitioning into χ_1 as

$$\exists \mathcal{P}(\gamma_i \in \chi_1 | \gamma_j \in \chi_2) > 0. \quad (7.11)$$

Finally, with Lemma 1 and the Markov chain properties, it indicates that the CND would filially transition into the recurrent states χ_1 with probability one:

$$\lim_{k \rightarrow +\infty} \mathcal{P}(\gamma_i \in \chi_1 | \gamma_j \in \chi_2) = 1, \quad (7.12)$$

which means that all states will be visited at some point. Given that all states in χ_2 will eventually lead to a state in χ_1 with probability one and that χ_1 contains the global optimal solutions, the proposed CND is thus proved to converge to the global optimal solution with the probability one. Theorem 1 is thus completely proved. ■

Theorem 2 Consider the interference of constant bias $\mathbf{w}(t) \in \mathbb{R}^n$; in the computation of GNSR model (7.4) for searching the global optimal solution, it is able to be stabilized. Meanwhile, its residual error $\lim_{t \rightarrow \infty} \|\rho(t)\|_2 = 0$, where $\|\cdot\|_2$ denotes the 2-norm.

Proof Since the exploration on the j subsystem of the GNSR model (7.4) could stack additively, we could easily use the similarity of symmetric matrices to get its j th subsystem as $\dot{\rho}_j = -(\varepsilon\hbar + \varsigma)\rho_j - \varsigma\varepsilon\hbar \int_0^t \rho_j(\tau) d\tau$ with $\hbar$ being the j th

eigenvalue of $\mathcal{M}(t)\mathcal{M}^T(t)$. Perturbed by constant bias $\mathbf{w}(t) \in \mathbb{R}^n$, its j th subsystem in the frequency domain by Laplace transformation is

$$\begin{aligned} s\rho_j(s) - \rho_j(0) &= -\varepsilon\hbar_j\rho_j(s) - \varsigma\rho_j(s) - \frac{\varsigma\varepsilon\hbar_j}{s}\rho_j(s) + w_j(s) \\ &= -\xi\rho_j(s) - \frac{\eta}{s}\rho_j(s) + w_j(s), \end{aligned} \quad (7.13)$$

that is,

$$\rho_j(s) = \frac{s(w_j(s) + \rho_j(0))}{s^2 + \xi s + \eta}, \quad (7.14)$$

where $\xi = \varepsilon\hbar_j + \varsigma > 0$ and $\eta = \varsigma\varepsilon\hbar_j > 0$. For this transfer function $\rho_j(s)$, its two poles $\mu_1 = (-\xi + \sqrt{\xi^2 - 4\eta})/2$ and $\mu_2 = (-\xi - \sqrt{\xi^2 - 4\eta})/2$ both remain in the left-half plane. By the stability theorem in the Laplace domain, the whole system $\rho_j(s)$ is stable. Therefore, based on the final value theorem (FVT) of the Laplace domain, we can get

$$\lim_{t \rightarrow \infty} \rho_j(t) = \lim_{s \rightarrow 0} s\rho_j(s) = \lim_{s \rightarrow 0} \frac{s^2(\rho_j(0) + w_j/s)}{s^2 + \xi s + \eta} = 0. \quad (7.15)$$

Consequently, $\lim_{t \rightarrow \infty} \|\boldsymbol{\rho}(t)\|_2 = 0$. The proof is completed. $\blacksquare$

Theorem 3 Consider the interference of linear scaling noise $\mathbf{w}(t) \in \mathbb{R}^n$; in the computation of GNSR model (7.4) for searching the global optimal solution, it is able to be stabilized. Meanwhile, its residual error $\lim_{t \rightarrow \infty} \|\boldsymbol{\rho}(t)\|_2 = \|\mathbf{w}\|_2/\eta$, where $\|\cdot\|_2$ denotes the 2-norm.

Proof With each element of $\mathbf{w}(t)$ being $w_j t$ and $w_j t \rightarrow w_j/s^2$, the j th perturbed subsystem in the frequency domain by Laplace transformation is

$$\begin{aligned} s\rho_j(s) - \rho_j(0) &= -\varepsilon\hbar_j\rho_j(s) - \varsigma\rho_j(s) - \frac{\varsigma\varepsilon\hbar_j}{s}\rho_j(s) + w_j/s^2 \\ &= -\xi\rho_j(s) - \frac{\eta}{s}\rho_j(s) + w_j/s^2, \end{aligned} \quad (7.16)$$

where $\xi = \varepsilon\hbar_j + \varsigma$; $\eta = \varsigma\varepsilon\hbar_j$. Furthermore, with FVT, it elicits

$$\lim_{t \rightarrow \infty} \rho_j(t) = \lim_{s \rightarrow 0} \frac{s^2(\rho_j(0) + w_j/s^2)}{s^2 + \xi s + \eta} = \frac{w_j}{\eta}. \quad (7.17)$$

Consequently, $\lim_{t \rightarrow \infty} \|\boldsymbol{\rho}(t)\|_2 = \|\mathbf{w}\|_2/\eta$. Thus, the proof is completed. $\blacksquare$

Theorem 4 Consider the interference of bounded unknown vector-form random noise $\boldsymbol{\varpi}(t) \in \mathbb{R}^n$; in the computation of GNSR model (7.4) for searching the

global optimal solution, it is able to be stabilized. Meanwhile, its residual error $\lim_{t \rightarrow \infty} \|\rho(t)\|_2 =$

$$\begin{cases} (2n \sup_{0 \leq \tau \leq t} |\varpi_j(\tau)|) / \sqrt{\xi^2 - 4\eta}, & \xi^2 - 4\eta > 0 \\ n(\varepsilon/\theta - 1/\vartheta_1) \sup_{0 \leq \tau \leq t} |\varpi_j(\tau)|, & \xi^2 - 4\eta = 0 \\ (4\eta n \sup_{0 \leq \tau \leq t} |\varpi_j(\tau)|) / (\xi \sqrt{\xi^2 - 4\eta}), & \xi^2 - 4\eta < 0. \end{cases} \quad (7.18)$$

Proof The j th random noise perturbed subsystem in the frequency domain by Laplace transformation is

$$\dot{\rho}_j(t) = -\xi \rho_j(t) - \eta \int_0^t \rho_j(\tau) d\tau + \delta_j(t), \quad (7.19)$$

of which the formulation of its solutions is divided into the three cases below.

- Case $\mathcal{A}$, if $\xi^2 - 4\eta > 0$, subsystem (7.19) could be solved as follows:

$$\begin{aligned} \rho_j(t) = & \frac{\rho_j(0)(\vartheta_1 \exp(\vartheta_1 t) - \vartheta_2 \exp(\vartheta_2 t))}{\vartheta_1 - \vartheta_2} + \frac{\int_0^t \vartheta_1 \exp(\vartheta_1(t - \tau)) \varpi_j(\tau) d\tau}{\vartheta_1 - \vartheta_2} - \\ & \frac{\int_0^t \vartheta_2 \exp(\vartheta_2(t - \tau)) \varpi_j(\tau) d\tau}{\vartheta_1 - \vartheta_2}, \end{aligned} \quad (7.20)$$

where $\vartheta_1 = (-\xi + \sqrt{\xi^2 - 4\eta})/2$ and $\vartheta_2 = (-\xi - \sqrt{\xi^2 - 4\eta})/2$. Next, based on the triangle inequality, it has

$$\begin{aligned} |\rho_j(t)| \leq & \frac{|\rho_j(0)(\vartheta_1 \exp(\vartheta_1 t) - \vartheta_2 \exp(\vartheta_2 t))|}{\vartheta_1 - \vartheta_2} + \frac{2}{\vartheta_1 - \vartheta_2} \max_{0 \leq \tau \leq t} |\delta_j(\tau)| \\ = & \frac{|\rho_j(0)(\vartheta_1 \exp(\vartheta_1 t) - \vartheta_2 \exp(\vartheta_2 t))|}{\vartheta_1 - \vartheta_2} + \frac{2}{\sqrt{\xi^2 - 4\eta}} \max_{0 \leq \tau \leq t} |\varpi_j(\tau)|. \end{aligned} \quad (7.21)$$

Then, extending (7.21) from the j subsystem to the whole system, we can acquire that

$$\lim_{t \rightarrow \infty} \|\rho(t)\|_2 \leq \frac{2n}{\sqrt{\xi^2 - 4\eta}} \sup_{0 \leq \tau \leq t} |\varpi_j(\tau)|. \quad (7.22)$$

- Case $\mathcal{B}$, if $\xi^2 - 4\eta = 0$, subsystem (7.19) could be solved as follows:

$$\begin{aligned} \rho_j(t) = & \rho_j(0) \exp(\vartheta_1 t) + \rho_j(0) \vartheta_1 t \exp(\vartheta_1 t) + \\ & \int_0^t (t - \tau) \vartheta_1 \exp(\vartheta_1(t - \tau)) \varpi_j(\tau) d\tau + \int_0^t \exp(\vartheta_1(t - \tau)) \varpi_j(\tau) d\tau. \end{aligned} \quad (7.23)$$

Under this circumstance, $\vartheta_{1,2} = (-\xi \pm \sqrt{\xi^2 - 4\eta})/2 = -\xi/2$. Besides, $\exists$ positive parameters ε and θ that meet with the inequality $|\vartheta_1|t \exp(\vartheta_1 t) \leq \varepsilon \exp(-\theta t)$. It elicits

$$|\rho_j(t)| \leq |\rho_j(0)\exp(\vartheta_1 t) + \rho_j(0)\vartheta_1 t \exp(\vartheta_1 t)| + \int_0^t |\varepsilon \exp(-\theta(t-\tau))\rho_j(\tau)|d\tau + \int_0^t |\exp(\vartheta_1(t-\tau))\rho_j(\tau)|d\tau. \quad (7.24)$$

Through further simplification, we can get

$$|\rho_j(t)| \leq |\rho_j(0)\exp(\vartheta_1 t) + \rho_j(0)\vartheta_1 t \exp(\vartheta_1 t)| + \left(\frac{\varepsilon}{\theta} - \frac{1}{\vartheta_1}\right) \max_{0 \leq \tau \leq t} |\varpi_j(\tau)|. \quad (7.25)$$

Utilizing FVT, the formulation in time domain could be

$$\lim_{t \rightarrow \infty} \|\rho(t)\|_2 \leq \left(\frac{\varepsilon}{\theta} - \frac{1}{\vartheta_1}\right) n \sup_{0 \leq \tau \leq t} |\varpi_j(\tau)|. \quad (7.26)$$

- Case $\mathcal{C}$, if $\xi^2 - 4\eta < 0$, subsystem (7.19) could be solved as follows:

$$\begin{aligned} \rho_j(t) = & \rho_j(0)\exp(at) \left(\frac{a \sin(bt)}{b} + \cos(bt) \right) + \\ & \int_0^t (a \sin(b(t-\tau)) \exp(a(t-\tau))/b \\ & + \cos(b(t-\tau)) \exp(a(t-\tau))) \varpi_j(\tau) d\tau, \end{aligned} \quad (7.27)$$

with $a = -\xi/2$, $b = \sqrt{4\eta - (\xi)^2}/2$. With the triangle inequality, it elicits

$$\begin{aligned} |\rho_j(t)| = & |\rho_j(0)\exp(at) \left(\frac{a \sin(bt)}{b} + \cos(bt) \right)| - \frac{\sqrt{a^2 + b^2}}{ab} \max_{0 \leq \tau \leq t} |\varpi_j(\tau)| \\ = & |\rho_j(0)\exp(at) \left(\frac{a \sin(bt)}{b} + \cos(bt) \right)| + \frac{4\eta_j}{\xi \sqrt{4\eta - \xi^2}} \max_{0 \leq \tau \leq t} |\varpi_j(\tau)|. \end{aligned} \quad (7.28)$$

Eventually, the conclusion can be reached as

$$\lim_{t \rightarrow \infty} \|\rho(t)\|_2 \leq \frac{4\eta_j n}{\xi_j \sqrt{4\eta_j - (\xi_j)^2}} \sup_{0 \leq \tau \leq t} |\varpi_j(\tau)|. \quad (7.29)$$

Thus, the proof is completed. ■

Remark 2 Various noises are frequently found in plenty of scientific applications and the noise formats including constant bias, linear scaling noise, and random noise are supported by the sound industrial background. An input step signal is intrinsically classified as the constant bias. The interference accumulated in the system could use the linear scaling noise as its formulation. Other interference inputs such as parametric variations, model uncertainties, and communication noise could be interpreted as the combination of these three abovementioned noises. The proofs through Theorems 2 to 4 deliver the strict theoretical analyses for stabilizing the proposed collaborative neural dynamics with noises, which is promising for plenty of practical requirements.

7.4 Simulation Verification

In this section, simulative examples and comparisons among other existing optimization algorithms are given to verify the effectiveness and robustness of the proposed collaborative neural dynamics (referred as CND) under different noisy environments.

Firstly, the number of agents $\mathcal{N}$, the maximum iteration number $k_{\max} = 1000$, and the loop termination condition are set as the same for all comparative methods when solving each benchmark function. For the proposed CND, the involved parameter includes the inertial weight $\mathfrak{w} = 0.6$, acceleration parameter $\lambda_1 = \lambda_2 = 2$, $\mathfrak{w}_1$ and $\mathfrak{w}_2$ scattered in $[0, 1]$ following the uniform distribution, and scaling factors $\varepsilon > 0$, $\varsigma > 0$. As proved in Theorem 4, the residual error of generated by GNSR model (7.5) is proportional to $\sup_{0 \leq \tau \leq t} |\varpi_j(\tau)|$ that symbolizes the upper bound of the absolute value of random noise $\varpi(t)$. Therefore, we could adjust scaling parameter ε and ς to increase term $\sqrt{\xi^2 - 4\eta}$ and $\xi\sqrt{\xi^2 - 4\eta}$ so as to lower the residual error. In the simulation, we usually set $\varepsilon = 10, 100, 1000$, and $\varsigma = 10, 1, 0.1$. The larger ε and ς are, the faster the convergence of GNSR (7.5) could be while the computation burden increases as well.

The parameter setting for comparative methods, i.e., combinational PSO (referred as CPSO) [28], projected neural network with PSO (referred as PNN) [33], recurrent neural network with PSO [23] (referred as RNN) [34] is provided as below. The inertial weight $\mathfrak{w} = 0.6$ (the range of CPSO [28] lies in $[0.4, 0.9]$), acceleration parameter $\lambda_1 = \lambda_2 = 2$, $\mathfrak{w}_1$, and $\mathfrak{w}_2$ scattered in $[0, 1]$ following the uniform distribution, which all are the same with the proposed CND. Separately, for RNN [34], it involves the scaling factor $\varepsilon > 0$. In the simulation, we usually set $\varepsilon = 10, 100, 1000$; for CPSO [28], normalization coefficient $\delta \in [0, 1]$. Furthermore, for other global optimization including PSO [23], GA [24], DE [27], MFO [35], and GWO [36], the values of parameters in the simulation follow the design specification outlined in the chapter.

All simulations are meticulously carried out on a high-performance computer running Windows 11, equipped with an Intel Core i7-12700 processor and 32 GB RAM, utilizing the MATLAB R2021a simulation software without external tool-boxes or packages.

7.4.1 Example I

Firstly, an illustrative nonlinear optimization example with time-varying constraints is given to testify the solving ability of GNSR model (7.4) with noise $\mathbf{w}(t) \in [0.9, 1.1]$. Other representative neural solvers such as PNN [33] and RNN [34] are chosen for comparison. Note that the following problem is with $\mathbf{u}(t) \in \mathbb{R}^{10}$, high time-variation, and strong nonlinearity. Besides, the scaling factors for GNSR model (7.4), PNN [33], and RNN [34] are the same with $\varepsilon = \varsigma = 10$. The whole time period is from 0 to 7 s.

$$\begin{aligned}
 \min F(\mathbf{u}(t), t) = & \\
 & (0.125 \sin(t) + 0.5)u_1^2(t) + (0.125 \cos(t) + 0.5)u_2^2(t) + \\
 & \sin(3t)u_1(t) + \cos(t)u_1(t)u_2(t)/2 + \cos(3t)u_2(t) + \\
 & 2u_3^2(t) + 2u_2(t)e^{-t}u_3(t) - \cos(t)u_3(t)u_4(t) + \\
 & u_4^2(t) + (1 + \cos(t))u_5^2(t) + 2u_6^2(t) - \\
 & e^{-t}u_5(t)u_6(t) + 2u_7^2(t) + u_1(t)u_7(t) + 2u_7(t) - \\
 & \sin(t)u_7(t)u_8(t) + u_8^2(t) + (u_9(t) - e^{-t})^2 + \\
 & \log(0.1t + 1)u_9(t)u_{10}(t) + (u_{10}(t) - t)^2,
 \end{aligned} \tag{7.30}$$

subject to

$$0.8 \sin(2t) = \sin(4t)u_1(t) + \cos(4t)u_2(t), \tag{7.31a}$$

$$t = u_8(t) + (t^3 - 4t^2 + t)u_9(t), \tag{7.31b}$$

$$-1 \leq u_1(t), \dots, u_6(t), u_8(t), \dots, u_{10}(t) \leq 4, \tag{7.31c}$$

$$-1 \leq u_7(t) \leq -0.1t. \tag{7.31d}$$

For time-varying equality constraints (7.31a)–(7.31b), with coefficient matrix $\mathcal{H}_1(t) = [\sin(4t), \cos(4t), \mathbf{0}_{1 \times 8}; \mathbf{0}_{1 \times 7}, 1, (t^3 - 4t^2 + t), 0]$, it elicits $\mathcal{H}_1(t)\mathbf{u}(t) = [0.8 \sin(2t); t]$. Similarly, for time-varying inequality constraints (7.31c)–(7.31d), with coefficient matrix $\mathcal{H}_2(t) = [I_{10}; -I_{10}]$, it elicits $\mathcal{H}_2(t)\mathbf{u}(t) = [4_{6 \times 1}; -0.1t; 4_{3 \times 1}; 1_{10 \times 1}]$.

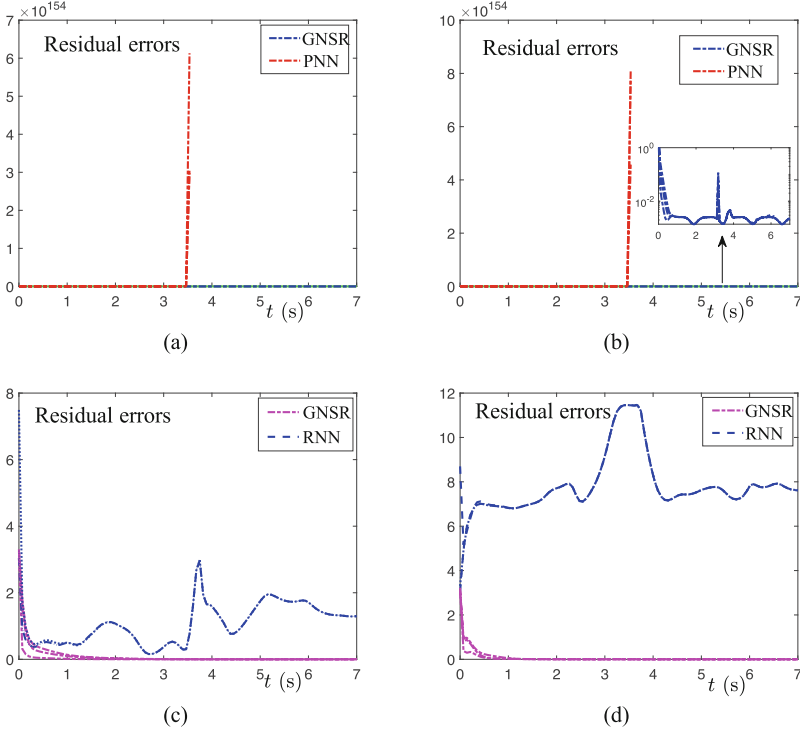
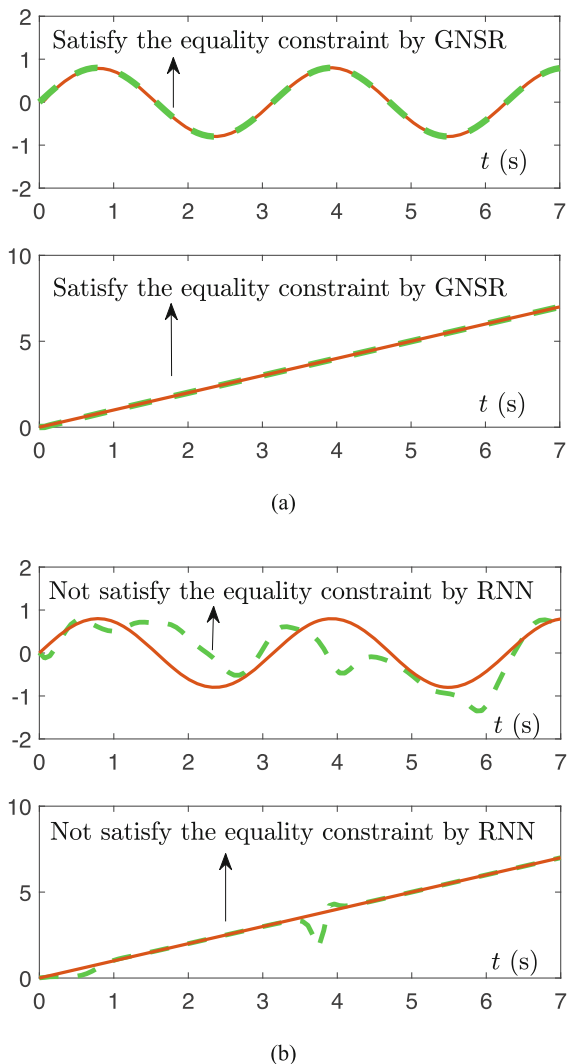


Fig. 7.1 Residual errors of GNSR model (7.4), PNN model [33], and RNN model [34] for handling time-varying optimization problem (7.30) with random initial values. (a) Comparison between GNSR model (7.4) and PNN model [33] without noise, (b) comparison between GNSR model (7.4) and PNN model [33] with random noise $w(t) \in [0.9, 1.1]$, (c) comparison between GNSR model (7.4) and RNN model [34] without noise, and (d) comparison between GNSR model (7.4) and RNN model [34] with random noise $w(t) \in [0.9, 1.1]$

Secondly, the residual errors of GNSR model (7.4) and PNN model [33] for handling time-varying optimization problem (7.30) with random initial values are plotted in Fig. 7.1. In detail, the noise-free case is in Fig. 7.1a. From the variation with time, it shows that residual errors of PNN model [33] diverge and largely deviate from zero while those of GNSR model (7.4) rapidly converge to zero. Under the noisy environment, the interference of random noise clearly enlarges residual errors of PNN model [33]. Differently, GNSR model (7.4) is capable of keeping residual errors from the disturbance of Gaussian random noise.

Then, Fig. 7.2 vividly presents the solving results of GNSR model (7.4) and RNN model [34] from the perspective of time-varying equality constraints (7.31a)–(7.31b). Specifically, it is evident in Fig. 7.2a that the output variable $u(t)$ obtained from GNSR model (7.4) completely satisfies two time-varying equality constraints (7.31a) and (7.31b). Then, from the counterpart, Fig. 7.2b synthesized by

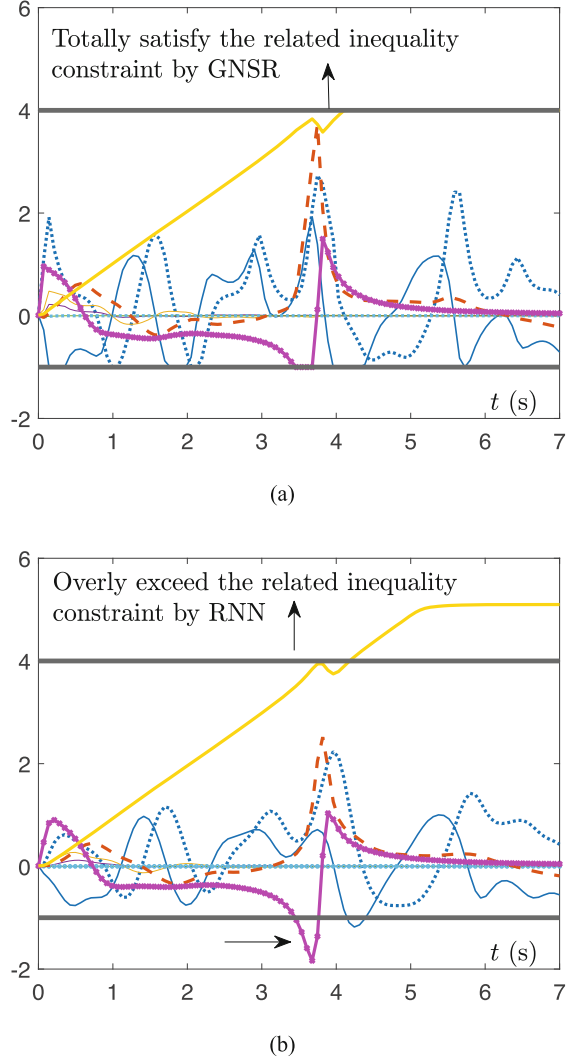
Fig. 7.2 Profiles of $\mathcal{H}_1(t)\mathbf{u}(t)$ and time-varying equality constraints (7.31a)–(7.31b) when GNSR model (7.4) and RNN model [34] handle time-varying optimization problem (7.30) without noise. (a) Profiles of GNSR model (7.4) and (b) profiles of RNN model [34]



RNN model [34], it is clearly shown that the output variable $\mathbf{u}(t)$ does not satisfy the corresponding time-varying equality constraints.

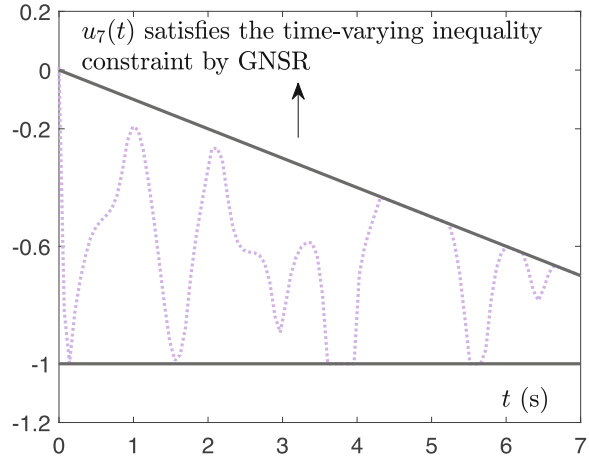
Next, Figs. 7.3 and 7.4 illustrate the performance of GNSR model (7.4) and RNN model [34] on dealing with the time-invariant inequality constraint (7.31c) and time-varying inequality constraint (7.31d), respectively. We can see through Figs. 7.3a to 7.4a that in the utilization of GNSR model (7.4), the output variable $\mathbf{u}(t)$ always keeps inside not only the time-invariant inequality constraint (7.31c) but the time-varying one (7.31d). However, as observed from Figs. 7.3b to 7.4b, the output variable $\mathbf{u}(t)$ drastically exceeds the bounds exerted by time-invariant inequality constraint (7.31c) as well as the time-varying one (7.31d).

Fig. 7.3 Profiles of $\mathcal{H}_2(t)\mathbf{u}(t)$ and time-invariant inequality constraints (7.31c) when GNSR model (7.4) and RNN model [34] handle time-varying optimization problem (7.30) without noise. (a) Profile of GNSR model (7.4) and (b) profile of RNN model [34]

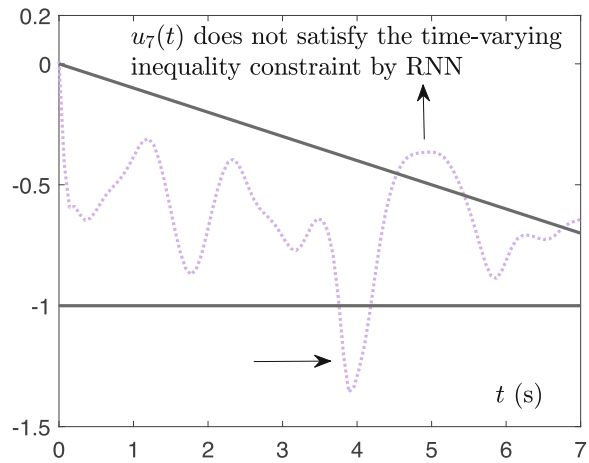


To sum up, these simulation results reveal that for time-varying optimization problems, for example, problem (7.30), PNN model [33] fails to meet the requirement of a dynamic solution when confronting high time variation, leaving large lagging errors and divergence states, let alone the consideration of disturbance during the solving process. In contrast, GNSR model (7.4) efficiently converges with residual error reaching zero and totally satisfies the time-varying constraints.

Fig. 7.4 Profiles of $\mathcal{H}_2(t)\mathbf{u}(t)$ and time-variant inequality constraints (7.31d) when GNSR model (7.4) and RNN model [34] handle time-varying optimization problem (7.30) without noise. (a) Profiles of GNSR model (7.4) and (b) profiles of RNN model [34]



(a)



(b)

7.4.2 Example II

To vividly illustrate the efficiency of the proposed CND with or without noises, we firstly use one classic multimodal problem, e.g., Ackley's function to test its performance as

$$\min F(\mathbf{u}) = -20 \exp \left(-\frac{1}{5} \sqrt{\frac{1}{n} \sum_{i=1}^n u_i^2} \right) + 20 + e^{-\exp \left(\frac{1}{n} \sum_{i=1}^n \cos(2\pi u_i) \right)}. \quad (7.32)$$

As seen from Fig. 7.5, there are plenty of locally optimal solutions near the global optimal solution $(0, 0, \dots, 0)$. Under the noise-free case, in Figs. 7.5a, 7.6a, and 7.7a, the collaborative neural dynamics efficiently solves Example II (7.32) with only 5 neural search agents during 9 iterations. Under the perturbation of random noise $\boldsymbol{\varpi}(t) \in [0.9, 1.1]$, it can be observed from Figs. 7.5b, 7.6b, and 7.7b that the collaborative neural dynamics gets rid of the instability caused by Gaussian random noise and finishes the optimization task with only 5 neural search agents during 11 iterations.

7.4.3 Simulation Comparisons

Table 7.2 lists 24 unimodal and multimodal benchmark functions, where coefficients for benchmark functions are provided as below. For benchmark function F_3 with $i = 1, \dots, 11$,

$$a_i = [0.1957, 0.1947, 0.1735, 0.1600, 0.0844, 0.0627, 0.0456, 0.0342, 0.0323, 0.0235, 0.0246];$$

$b_i = [0.25, 0.5, 1, 2, 4, 6, 8, 10, 12, 14, 16]$. For benchmark function F_{16} with $j = 1, \dots, 25$, $a_{1j} = [-32, -16, 0, 16, 32, 32, \dots, 0, 16, 32]$ and $a_{2j} = [-32, -32, -32, -32, -32, -16, \dots, 32, 32, 32]$. For benchmark function F_{20} and F_{21} with $i = 1, 2, 3, 4$ and $j = 1, 2, 3$,

$$\begin{cases} a_{ij} = [3, 10, 30; 0.1, 10, 35; 3, 10, 30; 0.1, 10, 35], \\ c_i = [1, 1.2, 3, 3.2], \\ p_{ij} = \begin{bmatrix} 0.3689 & 0.117 & 0.2673 \\ 0.4699 & 0.4387 & 0.7470 \\ 0.1091 & 0.8732 & 0.5547 \\ 0.03815 & 0.5743 & 0.8828 \end{bmatrix}. \end{cases}$$

For benchmark functions F_{22} , F_{23} , and F_{24} with $i = 1, \dots, 4$ and $j = 1, \dots, 10$, $c_j = [0.1, 0.2, 0.2, 0.4, 0.4, 0.6, 0.3, 0.7, 0.5, 0.5]$ and

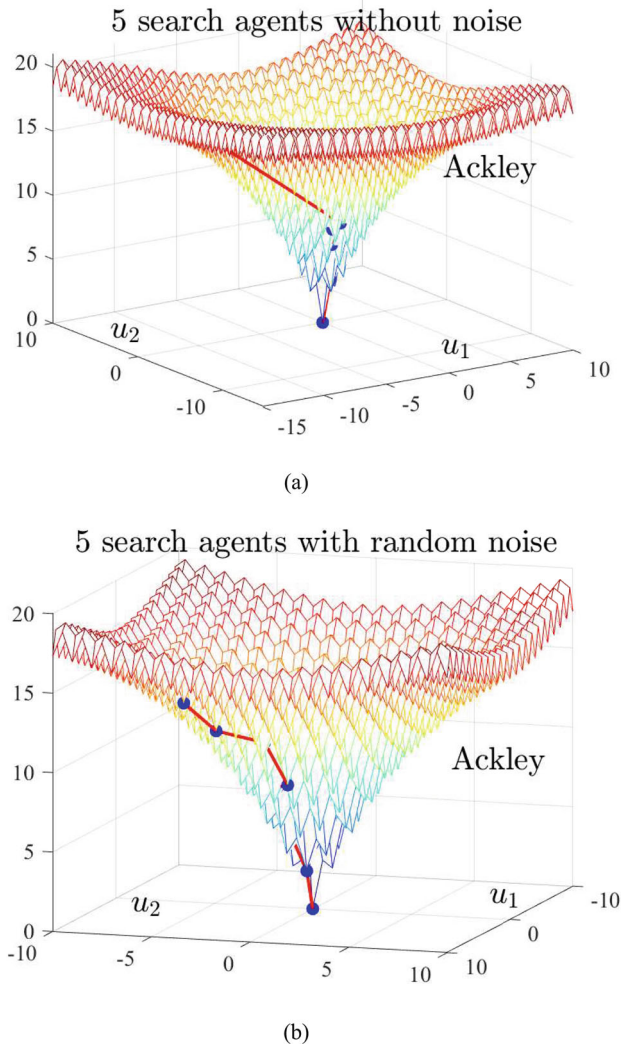
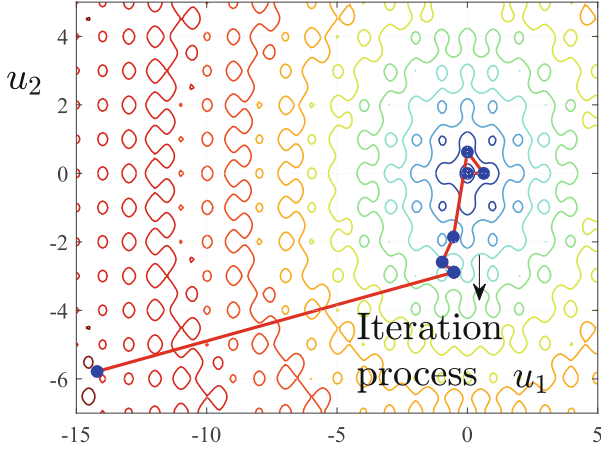
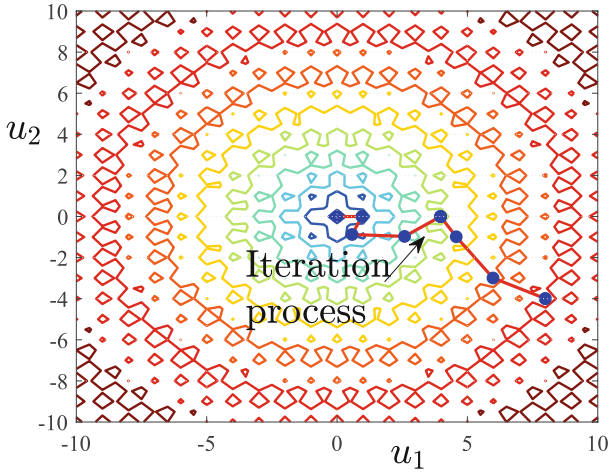


Fig. 7.5 The whole 3-D overview of the collaborative neural dynamics solving Example II (7.32) by 5 search agents under different circumstances. (a) Without noise and (b) with the perturbation of random noise $\varpi(t) \in [0.9, 1.1]$

$$a_{ij} = \begin{bmatrix} 4 & 1 & 8 & 6 & 3 & 2 & 5 & 8 & 6 & 7 \\ 4 & 1 & 8 & 6 & 7 & 9 & 5 & 1 & 2 & 3.6 \\ 4 & 1 & 8 & 6 & 3 & 2 & 3 & 8 & 6 & 7 \\ 4 & 1 & 8 & 6 & 7 & 9 & 3 & 1 & 2 & 3.6 \end{bmatrix}.$$



(a)



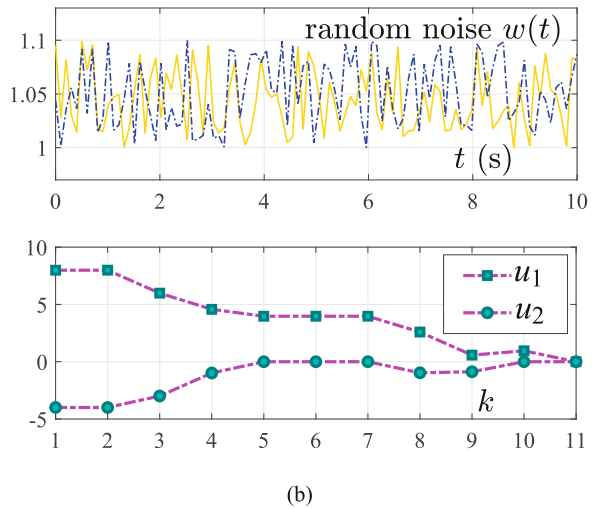
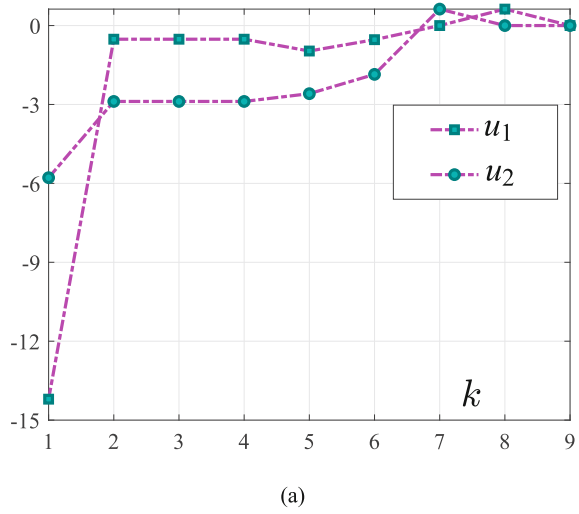
(b)

Fig. 7.6 The contour of Ackley's function and the iteration process of the collaborative neural dynamics solving Example II (7.32) by 5 search agents under different circumstances. (a) Without noise and (b) with the perturbation of random noise $\varpi(t) \in [0.9, 1.1]$

Besides, the corresponding theoretical optimal solutions are provided in Table 7.3, where “Dim” represents the dimension of the unknown vector $\mathbf{u}$; “Range” represents the lower and upper bounds of $\mathbf{u}$; “ $F_{\min}(\mathbf{u})$ ” represents the theoretical global optimal solution. Due to space limitation of the table, there are other theoretical optimal solutions:

1. $F_{3\min}(-0.1928, 0.1908, 0.1231, 0.1358) \approx 0.0003075$.

Fig. 7.7 Transient states of the global best solution (u_1, u_2) generated by the collaborative neural dynamics solving Example II (7.32) by 5 search agents under different circumstances. (a) Without noise and (b) with the perturbation of random noise $\varpi(t) \in [0.9, 1.1]$



2. $F_{10\min}(420.9687, \dots) = -12569.5$.
3. For functions F_{15} , there exist four local minimum points as

$$\begin{aligned}
 F_{15\min}(u_1, u_2) &= F(3.0, 2.0) = F(-2.805118, 3.131312) \\
 &= F(-3.779310, -3.283186) = F(3.584428, -1.848126) \\
 &= 0.
 \end{aligned}$$

4. For functions F_{17} , there exist two local minimum points as

Table 7.2 Benchmark functions for simulation verification

24 benchmark functions from F_1 to F_{24}
Schwefel function: $F_1(\mathbf{u}) = \sum_{i=1}^n u_i + \prod_{i=1}^n u_i $
Generalized Rosenbrock's function: $F_2(\mathbf{u}) = \sum_{i=1}^{n-1} [100(u_{i+1} - u_i^2)^2 + (u_i - 1)^2]$
Kowalik's function: $F_3(\mathbf{u}) = \sum_{i=1}^{11} [a_i - \frac{u_1(b_i^2 + b_i u_2)}{b_i^2 + b_i u_3 + u_4}]^2$
Sphere function: $F_4(\mathbf{u}) = \sum_{i=1}^n u_i^2$
Zakharov function: $F_5(\mathbf{u}) = \sum_{i=1}^n u_i^2 + (\sum_{i=1}^n 0.5iu_i)^2 + (\sum_{i=1}^n 0.5iu_i)^4$
Schwefel function: $F_6(\mathbf{u}) = \sum_{i=1}^n (\sum_{j=1}^i u_j)^2$
Schwefel function: $F_7(\mathbf{u}) = \max_i u_i , 1 \leq i \leq n$
Step function: $F_8(\mathbf{u}) = \sum_{i=1}^n ([u_i + 0.5])^2$
Quartic function i.e. Noise: $F_9(\mathbf{u}) = \sum_{i=1}^n iu_i^4 + \text{random}[0, 1)$
Generalized Schwefel's problem: $F_{10}(\mathbf{u}) = \sum_{i=1}^n -u_i \sin(\sqrt{ u_i })$
$F_{\min}(420.9687, \dots) = -12569.5$
Generalized Rastrigin's function: $F_{11}(\mathbf{u}) = \sum_{i=1}^n [u_i^2 - 10 \cos(2\pi u_i) + 10]$
Ackley's function: $F_{12}(\mathbf{u}) = -20 \exp(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^n u_i^2}) - \exp(\frac{1}{n} \sum_{i=1}^n \cos(2\pi u_i)) + 20 + e$
Griewank's function: $F_{13}(\mathbf{u}) = \frac{1}{4000} \sum_{i=1}^n u_i^2 - \prod_{i=1}^n \cos(\frac{u_i}{\sqrt{i}}) + 1$
Levy function: $F_{14}(\mathbf{u}) = \sin^2(\pi w_1) + \sum_{i=1}^{n-1} (w_i - 1)^2 [1 + 10 \sin^2(\pi w_i + 1)]$ $+ (w_n - 1)^2 [1 + \sin^2(2\pi w_n)], w_i = 1 + \frac{u_i - 1}{4}, \forall i = 1, \dots, n$
Himmelblau: $F_{15}(\mathbf{u}) = (u_1^2 + u_2 - 11)^2 + (u_1 + u_2^2 - 7)^2$
Shekel's foxhole function: $F_{16}(\mathbf{u}) = (\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^2 (u_i - a_{ij})^6})^{-1}$
Six-hump camelback function: $F_{17}(\mathbf{u}) = 4u_1^2 - 2.1u_1^4 + \frac{1}{3}u_1^6 + u_1u_2 - 4u_2^2 + 4u_2^4$
Branin function: $F_{18}(\mathbf{u}) = (u_2 - \frac{5.1}{4\pi^2}u_1^2 + \frac{5}{\pi}u_1 - 6)^2 + 10(1 - \frac{1}{8\pi}) \cos(u_1) + 10$
Goldstein-Price function: $F_{19}(\mathbf{u}) = [1 + (19 - 14u_1 + 3u_1^2 - 14u_2 + 6u_1u_2 + 3u_2^2)$ $(u_1 + u_2 + 1)^2] \times [30 + (2u_1 - 3u_2)^2$ $\times (18 - 32u_1 + 12u_1^2 + 48u_2 - 36u_1u_2 + 27u_2^2)]$
Hartman's family: $F_{20}(\mathbf{u}) = -\sum_{i=1}^4 c_i \exp(-\sum_{j=1}^3 a_{ij}(u_j - p_{ij})^2)$
Hartman's family: $F_{21}(\mathbf{u}) = -\sum_{i=1}^4 c_i \exp(-\sum_{j=1}^6 a_{ij}(u_j - p_{ij})^2)$
Shekel's family: $F_{22}(\mathbf{u}) = -\sum_{i=1}^5 [(u - a_i)(u - a_i)^T + c_i]^{-1}$
Shekel's family: $F_{23}(\mathbf{u}) = -\sum_{i=1}^7 [(u - a_i)(u - a_i)^T + c_i]^{-1}$
Shekel's family: $F_{24}(\mathbf{u}) = -\sum_{i=1}^{10} [(u - a_i)(u - a_i)^T + c_i]^{-1}$

Table 7.3 Theoretical optimal solutions of benchmark functions

24 benchmark functions from F_1 to F_{24}	Dim	Range	$F_{\min}(\mathbf{u})$
F_1	30	$[-10, 10]$	$F(0, \dots, 0) = 0$
F_2	30	$[-30, 30]$	$F(\mathbf{1}, \dots, \mathbf{1}) = \mathbf{0}$
F_3	4	$[-5, 5]$	$F_{3\min}$
F_4	30	$[-100, 100]$	$F(\mathbf{0}, \dots, \mathbf{0}) = \mathbf{0}$
F_5	30	$[-5, 10]$	$F(\mathbf{0}, \dots, \mathbf{0}) = \mathbf{0}$
F_6	30	$[-100, 100]$	$F(\mathbf{0}, \dots, \mathbf{0}) = \mathbf{0}$
F_7	30	$[-100, 100]$	$F(\mathbf{0}, \dots, \mathbf{0}) = \mathbf{0}$
F_8	30	$[-100, 100]$	$F(\mathbf{0}, \dots, \mathbf{0}) = \mathbf{0}$
F_9	30	$[-1.28, 1.28]$	$F(\mathbf{0}, \dots, \mathbf{0}) = \mathbf{0}$
F_{10}	30	$[-500, 500]$	$F_{10\min}$
F_{11}	30	$[-5.12, 5.12]$	$F(\mathbf{0}, \dots, \mathbf{0}) = \mathbf{0}$
F_{12}	30	$[-32, 32]$	$F(\mathbf{0}, \dots, \mathbf{0}) = \mathbf{0}$
F_{13}	2	$[-600, 600]$	$F(\mathbf{0}, \mathbf{0}) = \mathbf{0}$
F_{14}	30	$[-10, 10]$	$F(\mathbf{1}, \dots, \mathbf{1}) = \mathbf{0}$
F_{15}	2	$[-6, 6]$	$F(u_1, u_2) = \mathbf{0}$
F_{16}	2	$[-65, 65]$	$F(-32, 32) \approx \mathbf{1}$
F_{17}	2	$[-5, 5]$	$F_{17\min}$
F_{18}	2	$[-5, 5]$	$F_{18\min}$
F_{19}	2	$[-2, 2]$	$F(\mathbf{0}, -\mathbf{1}) = \mathbf{3}$
F_{20}	3	$[1, 3]$	$F_{20\min}$
F_{21}	6	$[0, 1]$	$F_{21\min}$
F_{22}	4	$[0, 10]$	$F(a_i) = -10.1532$
F_{23}	4	$[0, 10]$	$F(a_i) = -10.4028$
F_{24}	4	$[0, 10]$	$F(a_i) = -10.5363$

$$\begin{aligned}
 F_{17\min}(0.08983, -0.7126) &= F(-0.08983, 0.7126) \\
 &= -1.0316.
 \end{aligned}$$

5. For functions F_{18} , there exist three local minimum points as

6. $F_{20\min}(0.114, 0.556, 0.852) = -3.86$.

7. $F_{21\min}(0.201, 0.15, 0.477, 0.275, 0.311, 0.657) = -3.32$.

Furthermore, Tables 7.4, 7.5, and 7.6 present the simulation verification of the proposed CND and comparative methods including hybrid optimization methods: CPSO [28], PNN [33], RNN [34], and other nonconvex optimization methods: PSO [23], GA [24], DE [27], MFO [35], and GWO [36] with or without the presence of noise. Besides, we run the simulation of applying each comparative method to solve each benchmark function 40 times. Note that the iteration process is terminated only upon the fulfillment of condition (a) for three consecutive times or (c) in Sect. 7.2.2. When the matrix $\mathcal{M}$ becomes ill-conditioned, using the identity matrix as a replacement may be a method of avoiding numerical instability.

Table 7.4 Simulation results without the presence of noises

FLN	$F_1/10$		$F_2/50$	$F_3/2$
CND	Ave	1.73×10^{-6}	1.80×10^{-5}	3.0751×10^{-4}
	Std	4.21×10^{-7}	8.04×10^{-6}	2.2173×10^{-3}
PSO [25]	Ave	1.28×10^1	2.14×10^3	2.2173×10^{-3}
	Std	1.83	1.01×10^3	2.0907×10^{-3}
GA [26]	Ave	9.01×10^{-1}	1.88×10^2	2.0358×10^{-2}
	Std	1.96×10^{-1}	6.95×10^1	2.1017×10^{-2}
DE [29]	Ave	9.07×10^{-4}	4.62×10^1	—
	Std	9.29×10^{-5}	1.40×10^1	—
MFO [37]	Ave	3.00	4.59×10^3	4.2141×10^1
	Std	6.08	1.99×10^4	1.8535×10^1
GWO [38]	Ave	7.55×10^{-4}	2.65×10^1	5.3751×10^{-3}
	Std	1.01×10^{-4}	8.34×10^{-1}	1.5991×10^{-2}

Table 7.5 Simulation results with the presence of random noise [0, 11]

FLN	CND		RNN [36]	
	Ave	Std	Ave	Std
$F_4/(10)$	6.09×10^{-4}	2.22×10^{-4}	7.74×10^4	9.14×10^4
$F_5/(10)$	1.06×10^{-3}	1.38×10^{-6}	3.99×10^3	8.82×10^2
$F_6/(10)$	2.21×10^{-5}	2.25×10^{-5}	3.71×10^4	2.49×10^3
$F_7/(50)$	2.83×10^{-7}	3.31×10^{-8}	1.31×10^1	2.91
$F_8/(50)$	9.08×10^{-5}	1.25×10^{-4}	1.10×10^3	3.82×10^2
$F_9/(20)$	4.51×10^{-4}	4.30×10^{-5}	1.28×10^2	2.79×10^1
$F_{10}/(2)$	-1.26×10^4	1.09×10^{-1}	-4.12×10^3	5.45×10^2
$F_{11}/(50)$	2.35×10^{-4}	2.04×10^{-5}	4.28×10^2	2.45×10^1
$F_{12}/(10)$	2.19×10^{-6}	6.13×10^{-7}	9.59	1.28
$F_{13}/(2)$	2.82×10^{-5}	4.46×10^{-6}	2.16×10^{-1}	1.63×10^{-1}
$F_{14}/(30)$	1.30×10^{-6}	5.75×10^{-6}	2.16×10^2	3.70×10^2
$F_{15}/(2)$	5.07×10^{-6}	2.59×10^{-6}	1.64×10^2	1.35×10^2
$F_{16}/(2)$	1.0045	2.2373×10^{-3}	1.32×10^1	3.09
$F_{17}/(2)$	-1.0316	8.0680×10^{-5}	4.45×10^2	5.15×10^2
$F_{18}/(8)$	3.98×10^{-1}	1.42×10^{-5}	7.01	7.73
$F_{19}/(20)$	3.00	6.47×10^{-4}	6.37×10^2	2.61×10^2
$F_{20}/(3)$	-3.8627	1.5517×10^{-5}	-1.77	9.91×10^{-1}
$F_{21}/(6)$	-3.3137	1.4865×10^{-3}	-7.63×10^{-1}	5.89×10^{-1}
$F_{22}/(4)$	-10.1532	7.9483×10^{-6}	-2.95×10^{-1}	2.18×10^{-1}
$F_{23}/(4)$	-10.4029	5.7154×10^{-5}	-3.29×10^{-1}	1.37×10^{-1}
$F_{24}/(10)$	-10.5363	7.1331×10^{-5}	-5.06×10^{-1}	2.03×10^{-1}

Table 7.6 Simulation results with the presence of random noise [0, 11]

$F/\mathcal{N}$	PNN [35]		CPSO [30]	
	Ave	Std	Ave	Std
$F_4/(10)$	7.59×10^4	8.30×10^4	1.70×10^1	4.31
$F_5/(10)$	1.54×10^5	1.64×10^5	2.86×10^5	2.61×10^5
$F_6/(10)$	2.58×10^5	9.44×10^4	1.72×10^4	1.71×10^4
$F_7/(50)$	5.94×10^{-4}	3.25×10^{-4}	1.38	1.28×10^{-1}
$F_8/(50)$	2.57×10^3	1.50×10^1	1.05×10^1	2.14
$F_9/(20)$	1.29×10^2	2.91×10^1	1.28×10^2	2.25×10^1
$F_{10}/(2)$	-7.23×10^2	7.81×10^2	-7.35×10^3	2.29×10^3
$F_{11}/(50)$	4.34×10^2	1.86×10^1	4.36×10^2	2.07×10^1
$F_{12}/(10)$	5.87	2.14×10^{-1}	9.57×10^{-1}	1.59×10^{-1}
$F_{13}/(2)$	1.10	8.90×10^{-3}	3.65×10^{-2}	2.19×10^{-2}
$F_{14}/(30)$	2.30×10^2	3.22×10^1	2.20×10^2	3.24×10^1
$F_{15}/(2)$	1.16×10^2	1.07×10^2	1.55×10^2	1.89×10^2
$F_{16}/(2)$	4.31×10^2	1.30×10^2	4.51	5.25
$F_{17}/(2)$	2.61×10^2	3.70×10^2	3.75×10^2	5.53×10^2
$F_{18}/(8)$	8.16	7.55	5.04	3.06
$F_{19}/(20)$	1.19×10^4	2.56×10^4	9.54×10^3	2.23×10^4
$F_{20}/(3)$	-1.69	1.02	-1.68	9.84×10^{-1}
$F_{21}/(6)$	-8.28×10^{-1}	6.64×10^{-1}	-8.05×10^{-1}	4.63×10^{-1}
$F_{22}/(4)$	-2.46×10^{-1}	1.00×10^{-2}	-3.00×10^{-1}	2.35×10^{-1}
$F_{23}/(4)$	-3.91×10^{-1}	3.08×10^{-1}	-3.43×10^{-1}	1.92×10^{-1}
$F_{24}/(10)$	-4.29×10^{-1}	1.94×10^{-1}	-4.95×10^{-1}	2.42×10^{-1}

For the circumstance where comparative methods computing under the noise-free case, Table 7.4 and Fig. 7.8a illustrate results of using each method to solve benchmark functions F_1 , F_2 , and F_3 without noise during 40 repeated simulations. The proposed method generates the solution with higher accuracy compared with other methods. Besides, Tables 7.5, 7.6, and Fig. 7.8b show average iterations and optimal values when solving benchmark functions from F_4 to F_{24} with the presence of random noise [0, 11] during 40 repeated simulations. The proposed method is capable of achieving the optimal solution with a relatively small number of loop count and high accuracy while effectively mitigating the impact of measurement noise. The proposed method demonstrates robustness and accuracy in the presence of random noise, which is attributed to its ability to incorporate neural solution. Furthermore, the time complexity of the proposed CND is derived as $\mathcal{O}(k_{\max} \cdot \mathcal{N} \cdot n^3)$ based on the pseudocode in Algorithm 2. It is observed that the proposed method requires significantly fewer iterations than the other methods, achieving the global optimal solution in the first iteration and consistently finding it for the next two iterations. However, other methods required more iterations under the same parameter settings and some methods fail to find the optimal solution until the end of the iteration ($k = k_{\max} = 1000$). Furthermore, a notable limitation of metaheuristic

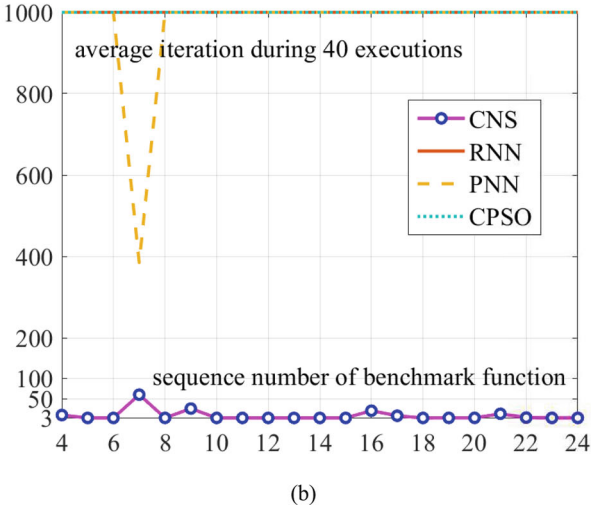
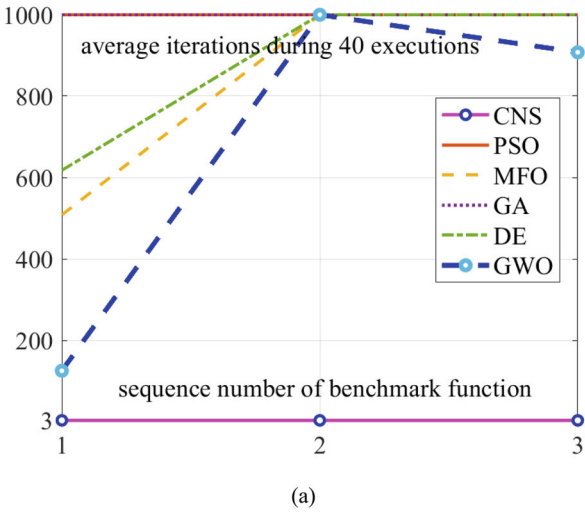


Fig. 7.8 The number of average iterations required by each method to reach the termination condition (a) or (c) (i.e., obtaining the optimal solution or reaching the maximum number of iterations) when solving benchmark functions in Table 7.2 during 40 repeated simulations. (a) Results of the proposed CNSD with nonconvex optimization methods without noise and (b) results of the proposed CNSD with hybrid optimization methods with the presence of random noise [0, 11]

approaches is their inability to assure the attainment of an optimal solution, particularly under the impact of perturbations, which significantly escalates their computational time complexity. In the future research, the computation time for each iteration of the proposed method can be significantly reduced by introducing upgraded hardware, for example, GPU parallel computing tools.

Overall, the simulation results presented above provide strong validation for the effectiveness and reliability of the proposed method in solving nonconvex optimization problems, even in the presence of noise. The proposed method efficiently and effectively optimizes the objective function, leveraging neural solution to mitigate the adverse effects of noise. Furthermore, the proposed method demonstrates superior performance compared to existing methods in terms of convergence speed, solution quality, and robustness. These results highlight the potential of the proposed method for a wide range of applications where the accurate and reliable generation of solutions are critical for achieving optimal performance.

7.5 Application to Manipulability Optimization of Redundant Manipulator

Avoiding singularity is a continuing concern within the control of the manipulator, since the singularity implies overlarge joint velocities or accelerations that may cause serious damage [37]. To this end, maximizing the manipulator's manipulability has been a pivotal index to determine the application feasibility. Nonetheless, how to effectively optimize manipulability while the robot manipulator conducts the specified task under the noisy environment is still in demand due to the fact that not only its objective function is nonconvex but the majority of control schemes have not considered the influence of different noises [38–43]. Therefore, in this part, the proposed collaborative neural dynamics is applied to maximize the manipulability with noises compared with representative methods for motion planning, and the mathematical problem is formulated as

$$\begin{aligned} \max F(\mathbf{u}(t)) &= \sqrt{\det(J(\mathbf{u}(t))J^T(\mathbf{u}(t)))}, \\ \text{s.t. } J(\mathbf{u}(t))\dot{\mathbf{u}}(t) &= \dot{\mathbf{r}}(t), \end{aligned} \quad (7.33)$$

of which $\mathbf{u} \in \mathbb{R}^6$ symbolizes the joint angle of the manipulator and $\dot{\mathbf{u}}$ stands for the joint velocity; function $F(\cdot)$ is time-varying and nonconvex; Jacobian matrix $J(\mathbf{u}(t)) \in \mathbb{R}^{3 \times 6}$; $\det(\cdot)$ represents the determinant operation; Cartesian coordinate $\mathbf{r}(t) \in \mathbb{R}^3$ and Cartesian velocity $\dot{\mathbf{r}}(t)$. The empirical evaluations of the proposed method on redundant manipulator are presented in Appendix B.

7.6 Conclusions

The research in this chapter focuses on proposing a new collaborative neural dynamics on noisy dynamic environments and the collaborative structure. New time-varying and robust improvements in neural dynamics method have been generated to collaborative neural dynamics for solving time-varying nonlinear optimization problems under noise-pollutant situations. The convergence issue arising from the metaheuristic method can be addressed by the cooperation of the neural dynamics method. Besides, theoretical analyses have proven that the proposed collaborative neural dynamics converges to the optimal solution of the nonconvex problem with the probability one. The application to manipulability optimization on redundant manipulators has added impetus on using the proposed scheme to various scientific problems. Promising directions for future research include exploring the integration of fuzzy control theory into the proposed method to enhance its flexibility for dealing with complex non-convex optimization problems and focusing on the obstacle avoidance performance on redundant manipulators to further enhance the applicability.

7.7 Appendix A

The following content illustrates how this GNSR (7.4) not only inherits the merit of gradient-based neural network (GNN) but offsets its insufficiency in handling the time-varying optimization. The definition of parameters could be found in Table 7.1. Firstly, with $\|\boldsymbol{\rho}(t)\|_2^2/2$ being the energy function, GNN is given as

$$\dot{\mathbf{u}}(t) = -\varepsilon \mathcal{M}^T(t) \boldsymbol{\rho}(t). \quad (7.34)$$

Secondly, a dynamical term $\mathbf{d}(t)$ is rendered to compensate for the lack of GNN in time-variant information as

$$\mathcal{M}(t) \dot{\mathbf{u}}(t) = -\varepsilon \mathcal{M}(t) \mathcal{M}^T(t) \mathfrak{F}(\boldsymbol{\rho}(t)) + \mathcal{M}(t) \mathbf{d}(t). \quad (7.35)$$

Next, one special case of the above Eq. (7.35) is that when $\boldsymbol{\rho}(t) = \mathbf{0}$, i.e., an equilibrium point is reached. Then, considering $\boldsymbol{\rho}(t) = \mathbf{0}$ in (7.35), the mathematical expression of $\mathbf{d}(t)$ could be obtained as

$$\mathcal{M}(t) \dot{\mathbf{u}}(t) = \mathcal{M}(t) \mathbf{d}(t). \quad (7.36)$$

Explicitly considering the derivative parameter of $\boldsymbol{\rho}(t)$, it elicits

$$\frac{d\boldsymbol{\rho}(t)}{dt} = \frac{\partial \boldsymbol{\rho}(t)}{\partial t} + \frac{\partial \boldsymbol{\rho}(t)}{\partial \mathbf{u}(t)} \frac{d\mathbf{u}(t)}{dt} = \frac{\partial \boldsymbol{\rho}(t)}{\partial t} + \mathcal{M}(t) \dot{\mathbf{u}}(t).$$

Then, in line with $\dot{\boldsymbol{\rho}}(t) = \mathbf{0}$ and formula (7.36), formula (7.35) is converted into

$$\mathcal{M}(t)\dot{\mathbf{u}}(t) = -\varepsilon\mathcal{M}(t)\mathcal{M}^T(t)\mathfrak{F}(\boldsymbol{\rho}(t)) - \frac{\partial\boldsymbol{\rho}(t)}{\partial t}. \quad (7.37)$$

Lastly, taking robustness improvement into consideration, we make another modification point into (7.37) as

$$\begin{cases} \mathbf{a}(t) = \boldsymbol{\rho}(t) + \varepsilon \int_0^t \mathcal{M}(\tau)\mathcal{M}^T(\tau)\mathfrak{F}(\boldsymbol{\rho}(\tau))d\tau \\ \dot{\mathbf{a}}(t) = -\varsigma\mathfrak{F}(\mathbf{a}(t)) \end{cases} \quad (7.38)$$

which equals to

$$\begin{aligned} \mathcal{M}(t)\dot{\mathbf{u}}(t) = & -\varepsilon\mathcal{M}(t)\mathcal{M}^T(t)\mathfrak{F}(\boldsymbol{\rho}(t)) - \partial\boldsymbol{\rho}(t)/\partial t - \\ & \varsigma\mathfrak{F}(\boldsymbol{\rho}(t) + \varepsilon\mathcal{M}(t)\mathcal{M}^T(t) \int_0^t \mathfrak{F}(\boldsymbol{\rho}(\tau))d\tau). \end{aligned} \quad (7.39)$$

The above formula (7.39) technically takes the form of an ordinary differential equation (ODE) and therefore any solver suitable for solving ODEs would be applicable. The highlight of GNSR (7.4) lies in the velocity compensation brought by the dynamical term $\mathbf{d}(t)$ and thus is capable of accurately solving the time-varying optimization problem and eliminating the large lagging error, which differs from those without the velocity compensation [44].

7.8 Appendix B

To supplement the empirical evaluations, the proposed method is applied to motion planning of redundant manipulators. The redundant manipulator, chosen as the platform due to its extra degrees of freedom that are increasingly significant in improving production efficiency and reducing manual labor, requires motion planning to determine how the robot should move or operate in complex environments [45, 46]. The proposed method works as the controller and generates joint velocities as control signals for the robot to realize the given motion planning task. The corresponding simulations are conducted based on a 7-DOF redundant manipulator named Franka Emika Panda [47] in the virtual robot experimentation platform (CoppeliaSim: <https://www.coppeliarobotics.com/coppeliaSim>). Then, physical experiments are conducted on a Franka Emika Panda robot. The desired trajectory for tracking is set as a circle curve with a radius of 0.08 m. The initial joint angles are $\mathbf{u} = [0, -0.7854, 0, -2.3562, 0, 1.5708, 0.7854]^T$ rad. The work frequency of Franka Emika Panda robotic platform is 1 kHz. The whole time-span lies in 10 s. To vividly show the results of the motion planning,

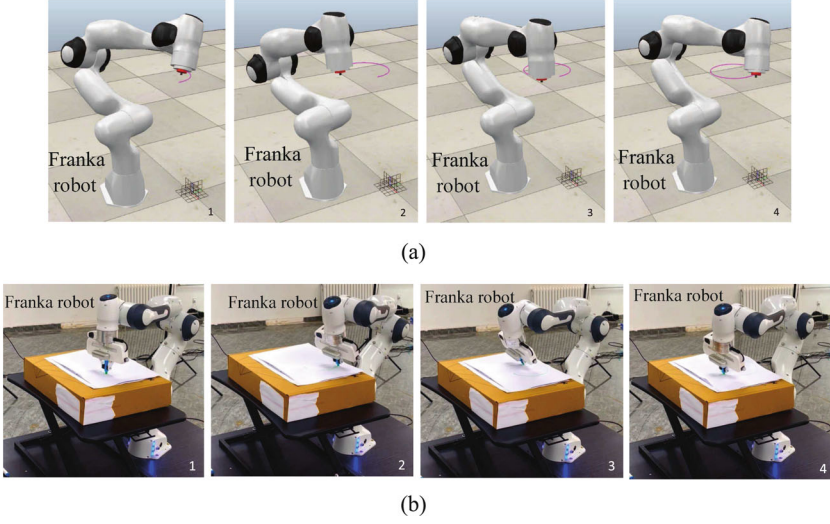


Fig. 7.9 Validation results of a circle trajectory generation of Franka Emika Panda robot, synthesized by the proposed method. (a) Snapshots of Franka Emika Panda robot working on CoppeliaSim. (b) Experiment results on Franka Emika Panda robot

a marker is attached to the end effector of the Franka Emika Panda robot in the physical experiment. As shown in Fig. 7.9, the end effector of the Franka Emika Panda robot driven by the proposed method smoothly finishes the circle trajectory. Snapshots of the simulation and experimental results correspond to Fig. 7.9a and b, respectively. The experimental result indicates that the proposed method is capable of solving real-world problems.

For a fair comparison, PUMA 560 manipulator is used for the tracking control with comparative schemes. As shown in Fig. 7.10, the end effector driven by the proposed collaborative neural dynamics smoothly finishes the circle trajectory tracking with randomly generated linear noise $w(t) = rand \times t, t \in [0, 50]s$, where *rand* represents the randomly generated coefficient for the linear noise. Although noise is increasing during the solving process, the position error from the actual path and the desired one is restricted under 10^{-5} m. Manipulability measure of the proposed collaborative neural dynamics ranges from 0.1 to 0.25, which not only illustrates that there is no singularity and the optimization results are rather effective. Besides, Fig. 7.10d shows the manipulability comparison of different methods [38–43] without noises conducting the same tracking task. It is obvious that the proposed collaborative neural dynamics is superior to these methods for manipulability optimization. This remarkable result indicates that the proposed collaborative neural dynamics would give its full play to plenty of scientific and engineering time-varying nonconvex problems with inherent tolerance to noises.

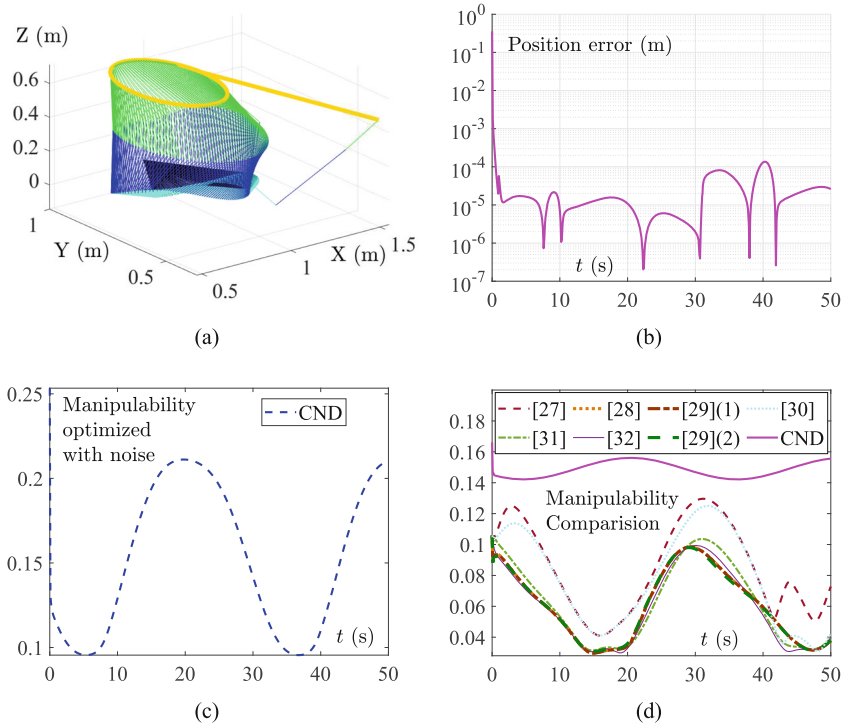


Fig. 7.10 Comparative performance of manipulability optimization (7.33) on PUMA 560 robot with noise $w(t) = rand \times t$. (a) The whole 3-D view. (b) Profiles of position errors. (c) Manipulability measure of the proposed method. (d) Manipulability comparisons of different methods without noise, where reference [40] contains two methods to be compared

References

1. Bao G, Zhang Y, Zeng Z (2020) Memory analysis for memristors and memristive recurrent neural networks. *IEEE/CAA J Autom Sin* 7(1):96–105
2. Chen J, Chen B, Zeng Z, Jiang P (2020) Event-triggered synchronization strategy for multiple neural networks with time delay. *IEEE Trans Cybern* 50(7):3271–3280
3. Yi M, Yang P, Chen M, Loc NT (2023) A DRL-driven intelligent joint optimization strategy for computation offloading and resource allocation in ubiquitous edge IoT systems. *IEEE Trans Emerg Top Comput Intell* 7(1):39–54
4. Chen J, Wang R, Wu D, Luo X (2023) A differential evolution-enhanced position-transitional approach to latent factor analysis. *IEEE Trans Emerg Top Comput Intell* 7(2):389–401
5. Pan Z, Wang L, Wang J, Lu J (2023) Deep reinforcement learning based optimization algorithm for permutation flow-shop scheduling. *IEEE Trans Emerg Top Comput Intell* 7(4):983–994
6. Yang Y, Chen Y, Lu Y (2017) A subspace conjugate gradient algorithm for large-scale unconstrained optimization. *Numer Algorithm* 76:813–828
7. Cotorruelo A, Ramirez DR, Limon D, Garone E (2021) Nonlinear MPC for tracking for a class of nonconvex admissible output sets. *IEEE Trans Autom Control* 66(8):3726–3732

8. Peng J, Yue S, Li H (2015) Equivalence: a phenomenon hidden among sparsity models minimization and minimization for information processing. *IEEE Trans Inf Theory* 61(7):4028–4033
9. Huang H, Yang C, Chen CLP (2021) Optimal robot-environment interaction under broad fuzzy neural adaptive control. *IEEE Trans Cybern* 51(7):3824–3835
10. Asim M, Wang Y, Wang K, Huang PQ (2020) A review on computational intelligence techniques in cloud and edge computing. *IEEE Trans Emerg Top Comput Intell* 4(6):742–763
11. Zhang YN, Zhang YY, Chen D, Xiao Z, Yan X (2017) From Davidenko method to Zhang dynamics for nonlinear equation systems solving. *IEEE Trans Syst Man Cybern Syst* 47(11):2817–2830
12. Zhang YY, Li S, Zhou X (2019) Recurrent-neural-network-based velocity-level redundancy resolution for manipulators subject to a joint acceleration limit. *IEEE Trans Ind Electron* 66(5):3573–3582
13. Hopfield J, Tank D (1985) ‘Neural’ computation of decisions in optimization problems. *Biol Cybern* 52:141–152
14. Liu N, Qin S (2019) A neurodynamic approach to nonlinear optimization problems with affine equality and convex inequality constraints. *Neural Netw* 109:147–158
15. Jia W, Qin S (2022) An adaptive penalty-like continuous-time algorithm to constrained distributed convex optimization. *J Frankl Inst* 359:3692–3716
16. Xia Y, Wang J (2004) A general projection neural network for solving monotone variational inequalities and related optimization problems. *IEEE Trans Neural Netw* 15(2):318–328
17. Liu Q, Wang J (2013) A one-layer projection neural network for nonsmooth optimization subject to linear equalities and bound constraints. *IEEE Trans Neural Netw Learn Syst* 24(5):812–824
18. Liu N, Wang J, Qin S (2022) A one-layer projection neural network for nonsmooth pseudo-convex optimization with quasiconvex inequality and affine equality constraints. *Neural Netw* 147:1–9
19. Liao B, Zhang YN, Jin L (2016) Taylor discretization of ZNN models for dynamic equality-constrained quadratic programming with application to manipulators. *IEEE Trans Neural Netw Learn Syst* 27(2):225–237
20. Li W, Ma X, Luo J, Jin L (2021) A strictly predefined-time convergent neural solution to equality- and inequality-constrained time-variant quadratic programming. *IEEE Trans Syst Man Cybern Syst* 51(7):4028–4039
21. Yazdani D, Cheng R, Yazdani D, Branke J, Jin Y, Yao X (2021) A survey of evolutionary continuous dynamic optimization over two decades-Part B. *IEEE Trans Evol Comput* 25(4):630–650
22. Dorigo M (1992) Optimization, learning and natural algorithms. Ph.D. thesis, Politecnico di Milano, Milano
23. Kennedy J, Eberhart R (1995) Particle swarm optimization. *Proc Int Joint Conf Neural Netw (IJCNN)* 4:1942–1948
24. Blackwell T, Branke J, Li X (2008) Swarm intelligence: introduction and applications, Heidelberg
25. Li Z, Zheng D, Hou H (2010) A hybrid particle swarm optimization algorithm based on nonlinear simplex method and Tabu search. In: *Proceedings of international symposium on neural networks*, Berlin, pp 126–135
26. Chen S-M, Chien C-Y (2011) Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques. *Expert Syst Appl* 38(12):14439–14450
27. Das S, Suganthan PN (2011) Differential evolution: a survey of the state-of-the-art. *IEEE Trans Evol Comput* 15(1):4–31
28. Jarbouli B, Cheikh M, Siarry P, Rebai A (2007) Combinatorial particle swarm optimization (CPSO) for partitional clustering problem. *Appl Math Comput* 192(2):337–345
29. Thangaraj R, Pant M, Abraham A, Bouvrya P (2011) Particle swarm optimization: hybridization perspectives and experimental illustrations. *Appl Math Comput* 217(12):5208–5226

30. Lu Q, Han Q-L, Liu S (2016) A cooperative control framework for a collective decision on movement behaviors of particles. *IEEE Trans Evol Comput* 20(6):859–873
31. Wei L, Jin L, Luo X (2024) A robust coevolutionary neural-based optimization algorithm for constrained nonconvex optimization. *IEEE Trans Neural Netw Learn Syst*. In press. <https://doi.org/10.1109/TNNLS.2022.3220806>
32. Gagniuc PA (2017) *Markov Chains: from theory to implementation and experimentation*. Wiley
33. Che H, Wang J (2019) A collective neurodynamic optimization approach to global and combinatorial optimization. *Neural Netw* 114:15–27
34. Yan Z, Fan J, Wang J (2017) A collective neurodynamic approach to constrained global optimization. *IEEE Trans Neural Netw Learn Syst* 28(5):1206–1215
35. Mirjalili S (2015) Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm. *Knowl-Based Syst* 89:228–249
36. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. *Adv Eng Softw* 69(1):46–61
37. Monchick L (2000) *Nonconvex optimization and its applications*. Springer, New York.
38. Jin L, Li S, La HM, Luo X (2017) Manipulability optimization of redundant manipulators using dynamic neural networks. *IEEE Trans Ind Electron* 64(6):4710–4720
39. Xia Y, Feng G, Wang J (2005) A primal-dual neural network for online resolving constrained kinematic redundancy in robot motion control. *IEEE Trans Syst Man Cybern Part B (Cybernetics)* 14(1):426–667
40. Li S, Zhang YN, Jin L (2017) Kinematic control of redundant manipulators using neural networks. *IEEE Trans Neural Netw Learn Syst* 28(10):2243–2254
41. Zhang YN, Yan X, Chen D, Guo D, Li Y (2016) QP-based refined manipulability-maximizing scheme for coordinated motion planning and control of physically constrained wheeled mobile redundant manipulators. *Nonlinear Dyn* 85(1):245–261
42. Cai B, Zhang YN (2012) Different-level redundancy-resolution and its equivalent relationship analysis for robot manipulators using gradientdescent and Zhang et al's neural-dynamic methods. *IEEE Trans Ind Electron* 59(8):3146–3155
43. Guo D, Zhang YN (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
44. Jin L, Zhang Y (2016) Continuous and discrete Zhang dynamics for real-time varying nonlinear optimization. *Numer Algorithms* 73(1):115–140
45. Qiu B, Guo J, Li X, Zhang Z, Zhang Y (2022) Discrete-time advanced zeroing neurodynamic algorithm applied to future equality-constrained nonlinear optimization with various noises. *IEEE Trans Cybern* 52(5):3539–3552
46. Yang M, Zhang Y, Zhang Z, Hu H (2020) Adaptive discrete ZND models for tracking control of redundant manipulator. *IEEE Trans Ind Inform* 16(12):7360–7368
47. Fan J, Jin L, Xie Z, Li S, Zheng Y (2022) Data-driven motion-force control scheme for redundant manipulators: a kinematic perspective. *IEEE Trans Ind Inform* 18(8):5338–5347

Glossary

Dual neural network A recurrent neural network is called dual neural network, if the network dynamic equation and implementation only use a group of dual decision variables.

Global convergence of a recurrent neural network A recurrent neural network is said to be globally convergent, if starting from any initial point $\hat{x}(t_0)$ taken in the whole associated Euclidean space, every state trajectory of the recurrent neural network converges to an equilibrium point $\hat{x}^*$ that depends on the initial state of the trajectory. Note that the initial time instant $t_0 \geq 0$, and there might exist many non-isolated equilibrium points $\hat{x}^*$.

Global exponential convergence of a recurrent neural network A recurrent neural network is said to be globally exponentially convergent if every trajectory starting from any initial point $\hat{x}(t_0)$ satisfies that, $\forall t \geq t_0 \geq 0$,

$$\|\hat{x}(t) - \hat{x}^*\| \leq \eta \|\hat{x}(t_0) - \hat{x}^*\| \exp(-\lambda(t - t_0)),$$

where η and λ are positive constants, $\hat{x}^*$ is an equilibrium point depending on initial state $\hat{x}(t_0)$, and the symbol $\|\cdot\|$ denotes the two-norm of a vector unless specified otherwise.

Kinematically redundant manipulator A kinematically redundant manipulator is a robot manipulator having more kinematic degrees of freedom (DOF) than required for its end effector's position and orientation.

Non-primal neural network A recurrent neural network is called non-primal neural network if the network dynamic equation and implementation use a group of auxiliary decision variables, in addition to using the original/primal variables or not.

Non-repetitive motion problem When the end effector traces a closed path in its workspace, the joint angles may not return to their initial values after completing the end-effector task.

Primal-dual neural network A recurrent neural network is called primal-dual neural network if the network dynamic equation and implementation use a group of dual decision variables, in addition to using the original/primal ones.

Primal neural network A recurrent neural network is called primal neural network if the network dynamic equation and implementation only use the original (or termed, primal) decision variables (here, the original/primal decision vector is $\hat{x}$).

Pseudoinverse-type solution It includes a minimum-norm particular solution and a homogeneous solution.

Stability of an equilibrium point An equilibrium point $\hat{x}^*$ of a recurrent neural network is locally stable if every trajectory, starting from the initial condition $\hat{x}(t_0)$ near the equilibrium point $\hat{x}^*$, stays near that equilibrium point $\hat{x}^*$. In addition, an equilibrium point $\hat{x}^*$ of a recurrent neural network is globally stable if every trajectory starting from any initial condition $\hat{x}(t_0)$ stays near that equilibrium point $\hat{x}^*$.