mathematics

MDPI

Article

Time-Varying Pseudoinversion Based on Full-Rank Decomposition and Zeroing Neural Networks

Hadeel Alharbi [1], Houssem Jerbi [2], Mourad Kchaou [3], Rabeh Abbassi [3], Theodore E. Simos [4,5,6,7,*], Spyridon D. Mourtas [8,9] and Vasilios N. Katsikis [8]

1 Department of Computer Science, College of Computer Science and Engineering, University of Hail, Hail 1234, Saudi Arabia
2 Department of Industrial Engineering, College of Engineering, University of Hail, Hail 1234, Saudi Arabia
3 Department of Electrical Engineering, College of Engineering, University of Hail, Hail 1234, Saudi Arabia
4 Department of Medical Research, China Medical University Hospital, China Medical University, Taichung 40402, Taiwan
5 Laboratory of Inter-Disciplinary Problems of Energy Production, Ulyanovsk State Technical University, 32 Severny Venetz Street, 432027 Ulyanovsk, Russia
6 Data Recovery Key Laboratory of Sichun Province, Neijing Normal University, Neijiang 641100, China
7 Section of Mathematics, Department of Civil Engineering, Democritus University of Thrace, 67100 Xanthi, Greece
8 Department of Economics, Mathematics-Informatics and Statistics-Econometrics, National and Kapodistrian University of Athens, Sofokleous 1 Street, 10559 Athens, Greece
9 Laboratory "Hybrid Methods of Modelling and Optimization in Complex Systems", Siberian Federal University, Prosp. Svobodny 79, 660041 Krasnoyarsk, Russia
* Correspondence: simost@susu.ru

Abstract: The computation of the time-varying matrix pseudoinverse has become crucial in recent years for solving time-varying problems in engineering and science domains. This paper investigates the issue of calculating the time-varying pseudoinverse based on full-rank decomposition (FRD) using the zeroing neural network (ZNN) method, which is currently considered to be a cutting edge method for calculating the time-varying matrix pseudoinverse. As a consequence, for the first time in the literature, a new ZNN model called ZNNFRDP is introduced for time-varying pseudoinversion and it is based on FRD. Five numerical experiments investigate and confirm that the ZNNFRDP model performs as well as, if not better than, other well-performing ZNN models in the calculation of the time-varying pseudoinverse. Additionally, theoretical analysis and numerical findings have both supported the effectiveness of the proposed model.

Keywords: pseudoinversion; dynamical system; full-rank decomposition; zeroing neural networks

MSC: 65F20; 15A24; 68T05

Citation: Alharbi, H.; Jerbi, H.; Kchaou, M.; Abbassi, R.; Simos, T.E.; Mourtas, S.D.; Katsikis, V.N. Time-Varying Pseudoinversion Based on Full-Rank Decomposition and Zeroing Neural Networks. *Mathematics* **2023**, *11*, 600. https://10.3390/math11030600

Academic Editor: Simeon Reich

Received: 22 December 2022
Revised: 13 January 2023
Accepted: 17 January 2023
Published: 24 January 2023

1. Introduction

The real-time solution to the matrix pseudoinverse (or Moore-Penrose inverse), which commonly occurs in robotics [1], nonlinear systems [2], game theory [3], and other engineering and science domains [4–6], has garnered a great deal of attention in recent decades. When a matrix A is taken into consideration, the pseudoinverse $A^{\dagger}$ is the generalization of the inverse matrix A^{-1}. For any matrix with entries that are real or complex numbers, the unique pseudoinverse is defined, and it is typically calculated using decomposition techniques, including the singular value decomposition, the QR decomposition, and the full-rank decomposition (FRD) [7,8]. To the best of our knowledge, these techniques typically targeted constant (or time-invariant) matrices. Nevertheless, real-time matrix pseudoinversion issues are frequently encountered. For instance, inverse kinematic issues for online control of redundant robot manipulators can be resolved through the online solution of the

time-varying pseudoinverse [9,10]. It is significant to note that the zeroing neural network (ZNN) method effectively enables the real-time solution of the time-varying pseudoinverse using both direct and indirect methods, such as matrix decomposition. For instance, when computing the time-varying pseudoinverse, a ZNN model based on singular value decomposition performs better than a direct ZNN model [11], so it is beneficial to look into ZNN models based on various matrix decomposition techniques. Therefore, the subject of this paper is the calculation of the time-varying pseudoinverse of a given time-varying matrix, based on the FRD method and the ZNN method. Our approach, in particular, complies with the concepts of Propositions 1 and 2, where Proposition 1 is based on [12] (Theorem D.4 (Full rank decomposition)) and Proposition 2 is restated from [12] [Theorem D.5] (Full rank decomposition and pseudoinversion). It is worth mentioning that, for $A \in \mathbb{R}^{m\times n}$, the calculation and representation of several generalized inverses is closely related to the next Penrose equations:

$$(i)\ AXA = A, \quad (ii)\ XAX = X, \quad (iii)\ (AX)^{\mathrm{T}} = AX, \quad (iv)\ (XA)^{\mathrm{T}} = XA \tag{1}$$

while the pseudoinverse is the matrix X that satisfies all four Penrose equations. Additionally, the following basic symbols from the paper are noteworthy: I_r implies the identity $r \times r$ matrix; $\mathbf{0}_{m\times n}$ and $\mathbf{1}_{m\times n}$ imply the zero and all-ones $m \times n$ matrices, respectively; $\mathrm{vec}(\cdot)$ implies the vectorization process; $\otimes$ implies the Kronecker product; $\dot{(\cdot)}$ implies the time derivative; $\|\cdot\|_F$ implies the matrix Frobenius norm. Last, the notations sum(.), zeros(.), eye(.), reshape(.), mod(.) and floor(.) in the algorithms of this paper follow the standard MATLAB function concept [13].

Proposition 1. *Let $A \in \mathbb{R}^{m\times n}$ be an arbitrary matrix of rank r, $B \in \mathbb{R}^{m\times n}$ be the reduced row echelon form of A, and $p \in \mathbb{N}^r$ be a vector that indicates the pivot columns of A. A unique FRD $A = CF$ is created from reduced row echelon form by setting the pivot columns of A in C, i.e., $C = A(:, p) \in \mathbb{R}^{m\times r}$, and the non-zero rows of B in F, i.e., $F = B(1 : r, :) \in \mathbb{R}^{r\times n}$.*

Proposition 2. *Let $A \in \mathbb{R}^{m\times n}$ be an arbitrary matrix of rank r, and $A = CF$ is the FRD of A, then:*

$$A^{\dagger} = F^{\mathrm{T}}(FF^{\mathrm{T}})^{-1}(C^{\mathrm{T}}C)^{-1}C^{\mathrm{T}}. \tag{2}$$

Many different approaches for computing the pseudoinverse have been developed over the past few decades, including fast calculation algorithms [14] and error bounds techniques [15]. However, when used for large-scale scenarios, these serial computational techniques (such as iterative algorithms) are ineffective. As matrix size rises, the computing workload increases significantly [16,17]. Thus, to decrease computational complexity and boost computational efficiency, a model with parallel computing capabilities is recommended. As effective parallel computing tools, recurrent neural networks (RNNs) have recently been thoroughly studied and used to address a variety of problems, such as mobile object localization [18], robotic motion tracking [19] and portfolio selection [20,21]. Gradient (or Hopfield) neural networks (GNNs), which are typical RNNs, have been heavily used to solve the constant matrix pseudoinversion problem in real time [22,23]. However, because it is difficult or perhaps impossible for GNNs and these serial methods to take into account the derivative information of the relevant matrices, neither serial methods nor GNNs are useful in calculating the time-varying matrix pseudoinverse.

The ZNN, a unique RNN with strong parallel processing capabilities, was introduced by Zhang et al. [24] to address the aforementioned problems. When it comes to computing the time-varying pseudoinverse, the ZNN approach is currently thought to be state-of-the-art. In particular, ZNNs were initially created to trace the time-varying inverse of a matrix [24]. Nevertheless, later versions of them were dynamic models for computing the time-varying pseudoinverse of full-row/column rank matrices [25–28]. Subsequent evolution has led to the evolvent of a great number of models for computing various generalized inverses [11], including the Drazin inverse [29], the ML-weighted pseudoinverse [30],

and outer inverses [31]. Along these, the ZNN method has been thoroughly investigated and employed to a broad variety of time-varying issues, with its main applications being the approximation of numerous matrix functions [32,33], problems involving quadratic optimization [10], and problems involving linear matrix equation systems [34,35]. To create ZNN dynamics, one must first define an error matrix, $E(t) \in \mathbb{R}^{m\times n}$, for the subjacent issue. The dynamical evolution is applied in the following phase:

$$\dot{E}(t) = \frac{\mathrm{d}E(t)}{\mathrm{d}t} = -\lambda \mathcal{H}(E(t)). \tag{3}$$

It is important to note that in (3) a design parameter $\lambda > 0$ is used to scale the convergence, and $\mathcal{H}(\cdot) : \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$ indicates the elementwise application of an increasing and odd activation function on the error matrix. In this work, we examine the ZNN evolution (3) by the linear activation function, leading to the next formula:

$$\dot{E}(t) = -\lambda E(t). \tag{4}$$

This paper proposes a ZNN model, dubbed ZNNFRDP, for the calculation of the time-varying pseudoinverse based on FRD.

These are the main contributions of this investigation:

- A new ZNN model, called ZNNFRDP, for the calculation of the time-varying pseudoinverse based on FRD is introduced and investigated. Keep in mind that ZNNFRDP stands for ZNN model based on FRD for the calculation of pseudoinverse.
- Theoretical analysis supported by five numerical experiments show that the ZNNFRDP model performs as well as, if not better than, other well-performing ZNN models in the computation of the time-varying pseudoinverse.

The remainder of the paper's layout is comprised of Sections 2–5. In particular, Section 2 defines and analyzes the ZNNFRDP model for computing the time-varying pseudoinverse, while Section 3 demonstrates and investigates the model's computational effectiveness. The concluding observations and inferences are given in Section 5.

2. Materials and Methods

In this section, the ZNN model, termed ZNNFRDP, that computes the time-varying pseudoinverse based on FRD is introduced and analyzed. Assume that $A(t) \in \mathbb{R}^{m\times n}$ is a differentiable time-varying matrix of constant rank r. In order to guarantee that the ZNNFRDP model applies FRD, the unique FRD presented in Proposition 1 is employed. In particular, consider that $p \in \mathbb{N}^r$ is a vector that indicates the pivot columns of $A(t)$. Then, creating $C(t) = A(:, p)(t) \in \mathbb{R}^{m\times r}$ and $F(t) \in \mathbb{R}^{r\times n}$ as indicating in Proposition 1, and setting $Y(t) = (C^{\mathrm{T}}(t)C(t))^{-1} \in \mathbb{R}^{r\times r}$, $Z(t) = (F(t)F^{\mathrm{T}}(t))^{-1} \in \mathbb{R}^{r\times r}$ and $X(t) = A^{\dagger}(t) \in \mathbb{R}^{n\times m}$ as indicating in proposition 2, the equations of the next group are satisfied in the case of pseudoinverse based on FRD:

$$\begin{cases} A(t) = C(t)F(t), \\ (C^{\mathrm{T}}(t)C(t))Y(t) = I_r, \\ Z(t)(F(t)F^{\mathrm{T}}(t)) = I_r, \\ X(t) = F^{\mathrm{T}}(t)Z(t)Y(t)C^{\mathrm{T}}(t). \end{cases} \tag{5}$$

The ZNNFRDP model takes into account determining the four unknowns $F(t)$, $Y(t)$, $Z(t)$ and $X(t)$ of (5). Notice that the elements below the major diagonal of $F(t)$ are zeros since it is a partitioned matrix of the reduced row echelon form of $A(t)$. Additionally, $Y(t)$ and $Z(t)$ are symmetric invertible matrices because $C(t)$ and $F(t)$ are full rank matrices. It is significant to note that when $A(t)$ is differentiable, $C(t)$ and $F(t)$ are also differentiable.

For zeroing (5), we set the next error matrix equations group (EMEG):

$$\begin{cases} E_1(t) = A(t) - C(t)F(t), \\ E_2(t) = (C^{\mathrm{T}}(t)C(t))Y(t) - I_r, \\ E_3(t) = Z(t)(F(t)F^{\mathrm{T}}(t)) - I_r, \\ E_4(t) = X(t) - F^{\mathrm{T}}(t)Z(t)Y(t)C^{\mathrm{T}}(t). \end{cases} \tag{6}$$

Furthermore, the first derivative of EMEG (6) is:

$$\begin{cases} \dot{E}_1(t) = & \dot{A}(t) - \dot{C}(t)F(t) - C(t)\dot{F}(t), \\ \dot{E}_2(t) = & (\dot{C}^{\mathrm{T}}(t)C(t) + C^{\mathrm{T}}(t)\dot{C}(t))Y(t) + (C^{\mathrm{T}}(t)C(t))\dot{Y}(t), \\ \dot{E}_3(t) = & \dot{Z}(t)(F(t)F^{\mathrm{T}}(t)) + Z(t)(\dot{F}(t)F^{\mathrm{T}}(t) + F(t)\dot{F}^{\mathrm{T}}(t)), \\ \dot{E}_4(t) = & \dot{X}(t) - \dot{F}^{\mathrm{T}}(t)Z(t)Y(t)C^{\mathrm{T}}(t) - F^{\mathrm{T}}(t)\dot{Z}(t)Y(t)C^{\mathrm{T}}(t) \\ & -F^{\mathrm{T}}(t)Z(t)\dot{Y}(t)C^{\mathrm{T}}(t) - F^{\mathrm{T}}(t)Z(t)Y(t)\dot{C}^{\mathrm{T}}(t). \end{cases} \tag{7}$$

When $\dot{E}(t)$ from (4) is replaced with $\dot{E}_i(t), i = 1, \ldots, 4$, defined in (7) and the equation is solved in terms of $\dot{F}(t), \dot{Y}(t), \dot{Z}(t), \dot{X}(t)$, the result below is obtained:

$$\begin{cases} -C(t)\dot{F}(t) = -\lambda E_1(t) - \dot{A}(t) + \dot{C}(t)F(t), \\ (C^{\mathrm{T}}(t)C(t))\dot{Y}(t) = -\lambda E_2(t) - (\dot{C}^{\mathrm{T}}(t)C(t) + C^{\mathrm{T}}(t)\dot{C}(t))Y(t), \\ \dot{Z}(t)(F(t)F^{\mathrm{T}}(t)) + Z(t)(\dot{F}(t)F^{\mathrm{T}}(t) + F(t)\dot{F}^{\mathrm{T}}(t)) = -\lambda E_3(t), \\ \dot{X}(t) - \dot{F}^{\mathrm{T}}(t)Z(t)Y(t)C^{\mathrm{T}}(t) - F^{\mathrm{T}}(t)\dot{Z}(t)Y(t)C^{\mathrm{T}}(t) - F^{\mathrm{T}}(t)Z(t)\dot{Y}(t)C^{\mathrm{T}}(t) \\ \quad = -\lambda E_4(t) + F^{\mathrm{T}}(t)Z(t)Y(t)\dot{C}^{\mathrm{T}}(t). \end{cases} \tag{8}$$

As shown below, the techniques of Kronecker product and vectorization are used to simplify the dynamic model of (8):

$$\begin{cases} -(I_n \otimes C(t))\mathrm{vec}(\dot{F}(t)) = \mathrm{vec}(-\lambda E_1(t) - \dot{A}(t) + \dot{C}(t)F(t)), \\ (I_r \otimes C^{\mathrm{T}}(t)C(t))\mathrm{vec}(\dot{Y}(t)) = \mathrm{vec}(-\lambda E_2(t) - (\dot{C}^{\mathrm{T}}(t)C(t) + C^{\mathrm{T}}(t)\dot{C}(t))Y(t)), \\ (F(t)F^{\mathrm{T}}(t) \otimes I_r)\mathrm{vec}(\dot{Z}(t)) + (F(t) \otimes Z(t))\mathrm{vec}(\dot{F}(t)) + (I_r \otimes Z(t)F(t))\mathrm{vec}(\dot{F}^{\mathrm{T}}(t)) \\ \quad = \mathrm{vec}(-\lambda E_3(t)), \\ \mathrm{vec}(\dot{X}(t)) - ((Z(t)Y(t)C^{\mathrm{T}}(t))^{\mathrm{T}} \otimes I_n)\mathrm{vec}(\dot{F}^{\mathrm{T}}(t)) - (C(t)Y^{\mathrm{T}}(t) \otimes F^{\mathrm{T}}(t))\mathrm{vec}(\dot{Z}(t)) \\ \quad -(C(t) \otimes F^{\mathrm{T}}(t)Z(t))\mathrm{vec}(\dot{Y}(t)) = \mathrm{vec}(-\lambda E_4(t) + F^{\mathrm{T}}(t)Z(t)Y(t)\dot{C}^{\mathrm{T}}(t)). \end{cases} \tag{9}$$

To create a straightforward dynamical model that can quickly calculate the $F(t)$, $Y(t)$, $Z(t)$, and $X(t)$, (9) must be simplified even more. As a consequence, the Lemma 1 based on [36] regarding vectorization is presented.

Lemma 1. *For $F \in \mathbb{R}^{r \times n}$, let $\mathrm{vec}(F) \in \mathbb{R}^{rn}$ denote the matrix F vectorization. What follows holds:*

$$\mathrm{vec}(F^{\mathrm{T}}) = Q\,\mathrm{vec}(F), \tag{10}$$

where $Q \in \mathbb{R}^{rn \times rn}$ be a fixed permutation matrix depended solely by the number of rows r and columns n of matrix F.

The next Algorithm 1 shows an algorithmic process for computing the permutation matrix Q in (10) that refers to a $r \times n$ matrix.

Algorithm 1 Matrix Q calculation.

Require: The number of columns n and rows r of matrix $F \in \mathbb{R}^{r \times n}$.

1: **procedure** PERMUT_MAT_Q(r, n)
2: Set c =eye(rn) and d =reshape($1 : rn, n, r$)
3: **return** $Q = c$(:,reshape($d^{\mathrm{T}}, 1, rn$))
4: **end procedure**

Ensure: The matrix Q.

Moreover, since all the elements below the major diagonal of $F(t)$ are zeros, just the elements of $\dot{F}(t)$ located in the major diagonal and above must be obtained. Therefore, it is significant to use a vector denoted $\dot{\mathbf{f}}(t)$ to replace $\dot{F}(t)$, by stacking these elements of $F(t)$ into $\dot{\mathbf{f}}(t)$. The dimension of (9) is lessened in this way, while $F(t)$ is forced to have all of its elements below the major diagonal zeros. Therefore, by using the $w = \sum_{i=n-r+1}^{n} i$ elements on and above the major diagonal of $F(t)$, it is possible to obtain the following equation that replaces $\mathrm{vec}(\dot{F}(t))$ in (9):

$$\mathrm{vec}(\dot{F}(t)) = G\dot{\mathbf{f}}(t), \tag{11}$$

where $G \in \mathbb{R}^{rn\times w}$ is an operational matrix which can be calculated applying the algorithmic procedure shown Algorithm 2. Further, the elements on and above the major diagonal of $\mathrm{vec}(\dot{F}(t))$ are piled to create the column vector $\dot{\mathbf{f}}(t) \in \mathbb{R}^{w}$.

Algorithm 2 Matrix G calculation.

Require: The number of columns n and rows r of matrix $F \in \mathbb{R}^{r\times n}$.

procedure OPE_MAT_G(r, n)
  Set w =sum$(n - r + 1 : n)$ and G =zeros(rn, w)
  for $q = 1 : rn$ **do**
    Set j =floor$(\frac{q-1}{r}) + 1$ and i =mod$(q - 1, r) + 1$
    if $i \leq j$ **then**
      Set $G(q, j+\mathrm{sum}(n - (1 : i - 1))(i > 1)) = 1$
    end if
  end for
  return G
end procedure

Ensure: The matrix G.

Additionally, because $Y(t)$ and $Z(t)$ are symmetric matrices, we only have to obtain their elements that are placed on and above its major diagonal. Thus, it is significant to use vectors denoted by $\dot{\mathbf{y}}(t)$ and $\dot{\mathbf{z}}(t)$ in place of $\dot{Y}(t)$ and $\dot{Z}(t)$, respectively, by stacking these elements of $Y(t)$ and $Z(t)$ into the vectors $\dot{\mathbf{y}}(t)$ and $\dot{\mathbf{z}}(t)$, respectively. The dimension of (9) is lessened in this way, while $Y(t)$ and $Z(t)$ are forced to be symmetric matrices. Therefore, using the $h = (r^2 + r)/2$ elements on and above the major diagonals of $Y(t)$ and $Z(t)$, it is possible to obtain the following equations that replace $\mathrm{vec}(\dot{Y}(t))$ and $\mathrm{vec}(\dot{Z}(t))$ in (9):

$$\mathrm{vec}(\dot{Y}(t)) = R\dot{\mathbf{y}}(t), \quad \mathrm{vec}(\dot{Z}(t)) = R\dot{\mathbf{z}}(t), \tag{12}$$

where $R \in \mathbb{R}^{r^2\times h}$ is an operational matrix which can be calculated applying the algorithmic procedure shown Algorithm 3. Further, the elements on and above the major diagonal of $\mathrm{vec}(\dot{Y}(t))$ and $\mathrm{vec}(\dot{Z}(t))$ are piled to create the column vectors $\dot{\mathbf{y}}(t), \dot{\mathbf{z}}(t) \in \mathbb{R}^{h}$.

Using the permutation matrix Q and the operational matrices G and R, (9) can be rewritten as follows:

$$\begin{cases} -(I_n \otimes C(t))G\mathrm{vec}\big(\dot{\mathbf{f}}(t)\big) = \mathrm{vec}\big(-\lambda E_1(t) - \dot{A}(t) + \dot{C}(t)F(t)\big), \\ (I_r \otimes C^{\mathrm{T}}(t)C(t))R\mathrm{vec}\big(\dot{\mathbf{y}}(t)\big) = \mathrm{vec}\big(-\lambda E_2(t) - (\dot{C}^{\mathrm{T}}(t)C(t) + C^{\mathrm{T}}(t)\dot{C}(t))Y(t)\big), \\ (F(t)F^{\mathrm{T}}(t) \otimes I_r)R\mathrm{vec}\big(\dot{\mathbf{z}}(t)\big) + (F(t) \otimes Z(t))G\mathrm{vec}\big(\dot{\mathbf{f}}(t)\big) + (I_r \otimes Z(t)F(t))QG\mathrm{vec}\big(\dot{\mathbf{f}}(t)\big) \\ \quad = \mathrm{vec}\big(-\lambda E_3(t)\big), \\ \mathrm{vec}\big(\dot{X}(t)\big) - ((Z(t)Y(t)C^{\mathrm{T}}(t))^{\mathrm{T}} \otimes I_n)QG\mathrm{vec}\big(\dot{\mathbf{f}}(t)\big) - (C(t)Y^{\mathrm{T}}(t) \otimes F^{\mathrm{T}}(t))R\mathrm{vec}\big(\dot{\mathbf{z}}(t)\big) \\ \quad -(C(t) \otimes F^{\mathrm{T}}(t)Z(t))R\mathrm{vec}\big(\dot{\mathbf{y}}(t)\big) = \mathrm{vec}\big(-\lambda E_4(t) + F^{\mathrm{T}}(t)Z(t)Y(t)\dot{C}^{\mathrm{T}}(t)\big). \end{cases} \tag{13}$$

Then, by setting

$$
\begin{aligned}
&k_1(t) = -(I_n \otimes C(t))G, \quad k_2(t) = (I_r \otimes C^{\mathrm{T}}(t)C(t))R, \quad k_4(t) = (F(t)F^{\mathrm{T}}(t) \otimes I_r)R,\\
&k_3(t) = (F(t) \otimes Z(t))G + (I_r \otimes Z(t)F(t))QG, \quad k_5(t) = -((Z(t)Y(t)C^{\mathrm{T}}(t))^{\mathrm{T}} \otimes I_n)QG,\\
&k_6(t) = -(C(t) \otimes F^{\mathrm{T}}(t)Z(t))R, \quad k_7(t) = -(C(t)Y^{\mathrm{T}}(t) \otimes F^{\mathrm{T}}(t))R,\\
&\mathbf{K}(t) = \begin{bmatrix} k_1(t) & \mathbf{0}_{mn\times h} & \mathbf{0}_{mn\times h} & \mathbf{0}_{mn\times mn} \\ \mathbf{0}_{r^2\times w} & k_2(t) & \mathbf{0}_{r^2\times h} & \mathbf{0}_{r^2\times mn} \\ k_3(t) & \mathbf{0}_{r^2\times h} & k_4(t) & \mathbf{0}_{r^2\times mn} \\ k_5(t) & k_6(t) & k_7(t) & I_{mn} \end{bmatrix}, \quad \mathbf{a}(t) = \begin{bmatrix} \mathrm{vec}(F(t)) \\ \mathrm{vec}(Y(t)) \\ \mathrm{vec}(Z(t)) \\ \mathrm{vec}(X(t)) \end{bmatrix},\\
&\dot{\mathbf{a}}(t) = \begin{bmatrix} \mathrm{vec}(\dot{F}(t)) \\ \mathrm{vec}(\dot{Y}(t)) \\ \mathrm{vec}(\dot{Z}(t)) \\ \mathrm{vec}(\dot{X}(t)) \end{bmatrix}, \quad \mathbf{b}(t) = \begin{cases} \mathrm{vec}(-\lambda E_1(t) - \dot{A}(t) + \dot{C}(t)F(t)), \\ \mathrm{vec}(-\lambda E_2(t) - (\dot{C}^{\mathrm{T}}(t)C(t) + C^{\mathrm{T}}(t)\dot{C}(t))Y(t)), \\ \mathrm{vec}(-\lambda E_3(t)), \\ \mathrm{vec}(-\lambda E_4(t) + F^{\mathrm{T}}(t)Z(t)Y(t)\dot{C}^{\mathrm{T}}(t)), \end{cases}
\end{aligned} \tag{14}
$$

the ZNN model below, termed ZNNFRDP, is proposed for calculating the time-varying pseudoinverse based on FRD of an arbitrary input matrix $A(t) \in \mathbb{R}^{m\times n}$:

$$\mathbf{K}^{\mathrm{T}}(t)\mathbf{K}(t)\dot{\mathbf{a}}(t) = \mathbf{K}^{\mathrm{T}}(t)\mathbf{b}(t), \tag{15}$$

which can be solved successfully in MATLAB using an `ode` solver. Note that $\dot{\mathbf{a}}(t), \mathbf{a}(t) \in \mathbb{R}^{(w+2h+mn)}$, $\mathbf{b}(t) \in \mathbb{R}^{(2r^2+2mn)}$ and $\mathbf{K}(t) \in \mathbb{R}^{(2r^2+2mn)\times(w+2h+mn)}$, while $\mathbf{K}^{\mathrm{T}}(t)\mathbf{K}(t) \in \mathbb{R}^{(w+2h+mn)\times(w+2h+mn)}$ is a non-singular mass matrix. In accordance with Theorem 1, the ZNNFRDP model (15) converges to the theoretical solution (TSol) of the time-varying pseudoinverse.

Algorithm 3 Matrix R calculation.

Require: A real symmetric matrix's column or row number r.

procedure OPE_MAT_R(r)
  Put $h = (r^2 + r)/2$ and R =zeros(r^2, h)
  for $w = 1 : r^2$ **do**
    Set d =mod$(w - 1, r) + 1$ and c =floor$(\frac{w-1}{r}) + 1$
    if $c \geq d$ **then**
      Put $R\left(w, d + c\frac{c-1}{2}\right) = 1$
    else
      Put $R\left(w, c + d\frac{d-1}{2}\right) = 1$
    end if
  end for
  return R
end procedure

Ensure: The matrix R.

Theorem 1. *Let $A(t) \in \mathbb{R}^{m\times n}$ be a differentiable matrix of rank r and $p \in \mathbb{N}^r$ be a vector that contains the numbers of all pivot columns of $A(t)$. Starting from any initial price $\mathbf{a}(0)$, the ZNNFRDP model (15) exponentially converges to the TSol $\mathbf{a}^*(t)$ at each time $t \in [0, t_f) \subseteq [0, +\infty)$. Furthermore, the TSol of the time-varying pseudoinverse is the last mn elements of $\mathbf{a}^*(t)$.*

Proof. The EMEG is declared as in (6) in order to determine the TSol of the time-varying pseudoinverse. The model (8) is developed using the linear ZNN design (4) for zeroing (6). When $t \to \infty$, every equation in the EMEG (8) converges to the TSol starting from any initial price, according to [24] [Theorem 1]. As a result, the ZNNFRDP model (15) also converges to the TSol $\mathbf{a}^*(t)$ starting from any initial price $\mathbf{a}(0)$ when $t \to \infty$ since it is really an alternate version of (8). In accordance with (14), the TSol of the time-varying pseudoinverse is the last mn elements of $\mathbf{a}^*(t)$. Therefore, the proof is finished. □

Additionally, diagram presentation can be used to describe the above-mentioned ZNNFRDP model process for real-time solution of the time-varying pseudoinverse of matrix $A(t)$. For the block diagram presentation, the dynamical Equation (15) can be transformed as follows:

$$\dot{\mathbf{a}}(t) = (I_{(w+2h+mn)} - \mathbf{K}^{\mathrm{T}}(t)\mathbf{K}(t))\dot{\mathbf{a}}(t) + \mathbf{K}^{\mathrm{T}}(t)\mathbf{b}(t), \tag{16}$$

and the corresponding block diagram of the ZNNFRDP model (16) is shown in Figure 1.

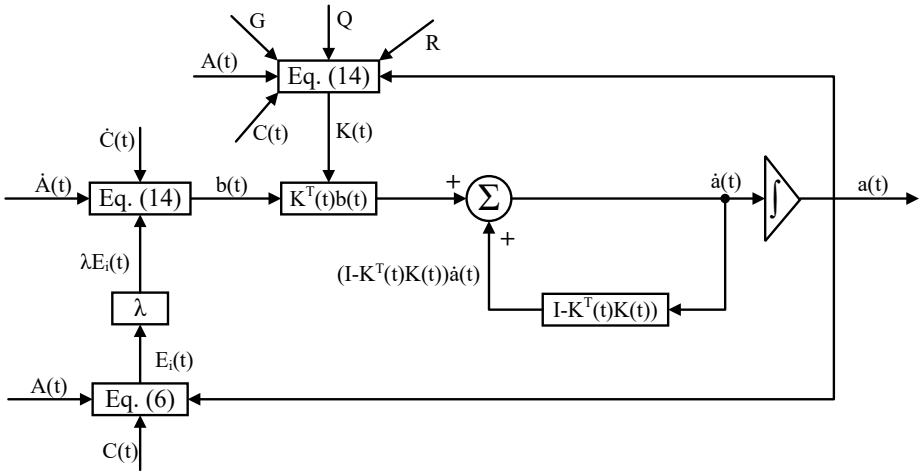


Figure 1. Block diagram of the ZNNFRDP model for real-time solution of the time-varying pseudoinverse of matrix $A(t)$.

Lastly, the complexity of producing and solving (16) contributes to the ZNNFRDP model's overall computational complexity. The computational complexity of producing (16) is $\mathcal{O}((w+2h+mn)^2)$ operations since each iteration of the equation has $(w+2h+mn)^2$ multiplication and $w+2h+mn$ addition operations. Considering that a linear system of equations is solved at each step by an implicit solver such as `ode15s`, the complexity of solving (16) is $\mathcal{O}((w+2h+mn)^3$ because it involves a $(w+2h+mn) \times (w+2h+mn)$ matrix. Therefore, the ZNNFRDP model's overall computational complexity is $\mathcal{O}((w+2h+mn)^3)$.

3. Results

The performance of the ZNNFRDP (15) model in five numerical experiments (NEs), which involve calculating time-varying pseudoinverses of rank deficient matrices, is examined in this section. Furthermore, the ZNNFRDP model is compared to two additional ZNN models, namely ZNNP and ZNNSVDP, which are presented in [11]. It is worth mentioning that both the ZNNP and ZNNSVDP models calculate the pseudoinverse of an arbitrary matrix, with the ZNNSVDP model employing singular value decomposition to do so. With the exception of $X(0)$, where all-ones matrices have been set, the initial prices have been set to matrices with ones on the major diagonal and elsewhere zeros for $F(0), Y(0)$ and $Z(0)$ in all NEs. Furthermore, in the denotation of the input matrix, the functions $\eta(t) = \cos(t)$ and $\theta(t) = \sin(t)$ have been used. Last, the MATLAB solver `ode15s` is employed for the computations in all NEs within the time range $[0, 10]$ and with the parameter $\lambda = 10$. Because of this, all ZNN models find the real-time solution of the time-varying pseudoinverse of a given matrix $A(t)$ starting at $t = 0$ and ending at $t = 10$.

3.1. Experiment 1

Assume the following 3×3 input matrix of rank 2 with $p = [1, 2]$:

$$A(t) = \begin{bmatrix} 3+\eta(2t) & \theta(t)+1 & \theta(t)+1 \\ 2+\theta(2t) & \eta(t)-20 & \eta(t)-20 \\ 2+\theta(2t) & \eta(t)-20 & \eta(t)-20] \end{bmatrix}.$$

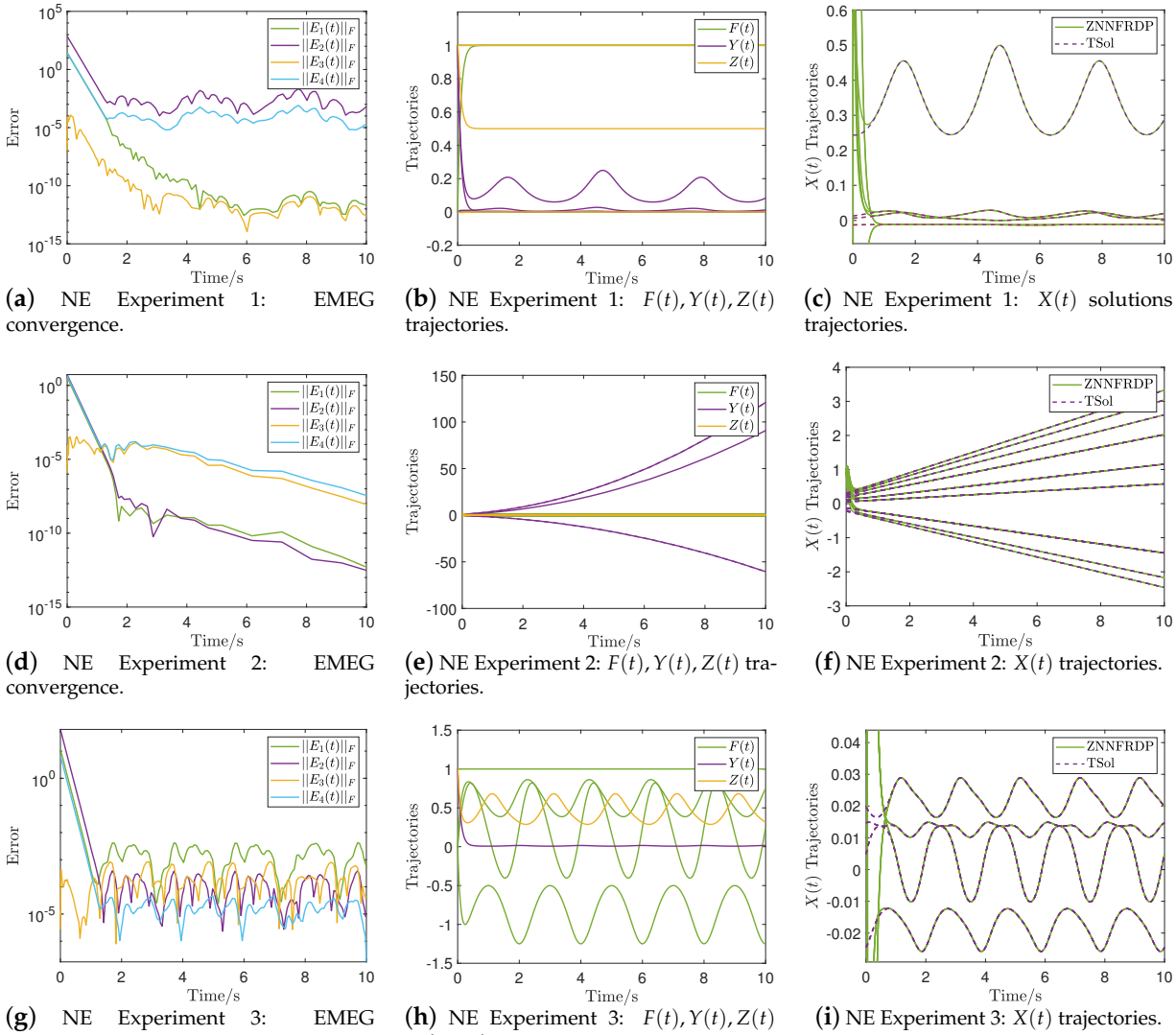
Presuming that $C(t) = A(:, p)(t)$, the findings of the ZNNFRDP model are displayed in Figures 2a–c and 3a,b.

3.2. Experiment 2

Assume the following 4×7 input matrix of rank 3 with $p = [1, 2, 4]$:

$$A(t) = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \odot \frac{1}{t+1}.$$

Presuming that $C(t) = A(:, p)(t)$, the findings of the ZNNFRDP model are displayed in Figures 2d–f and 3c,d.

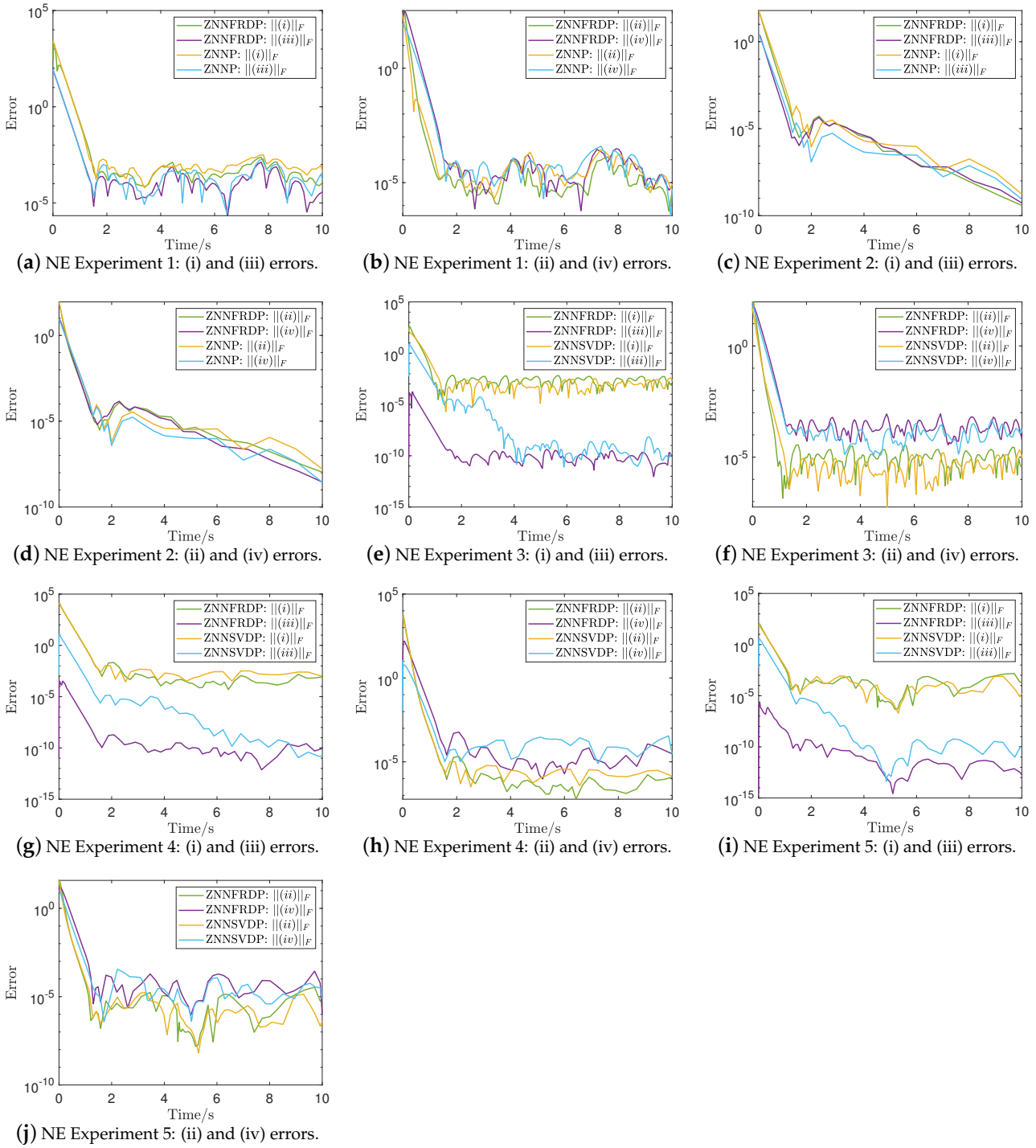


(a) NE Experiment 1: EMEG convergence. **(b)** NE Experiment 1: $F(t), Y(t), Z(t)$ trajectories. **(c)** NE Experiment 1: $X(t)$ solutions trajectories.

(d) NE Experiment 2: EMEG convergence. **(e)** NE Experiment 2: $F(t), Y(t), Z(t)$ trajectories. **(f)** NE Experiment 2: $X(t)$ trajectories.

(g) NE Experiment 3: EMEG convergence. **(h)** NE Experiment 3: $F(t), Y(t), Z(t)$ trajectories. **(i)** NE Experiment 3: $X(t)$ trajectories.

Figure 2. EMEG convergence and solutions' trajectories of the ZNNFRDMP model in NEs in Sections 3.1–3.3.



(a) NE Experiment 1: (i) and (iii) errors. **(b)** NE Experiment 1: (ii) and (iv) errors. **(c)** NE Experiment 2: (i) and (iii) errors.

(d) NE Experiment 2: (ii) and (iv) errors. **(e)** NE Experiment 3: (i) and (iii) errors. **(f)** NE Experiment 3: (ii) and (iv) errors.

(g) NE Experiment 4: (i) and (iii) errors. **(h)** NE Experiment 4: (ii) and (iv) errors. **(i)** NE Experiment 5: (i) and (iii) errors.

(j) NE Experiment 5: (ii) and (iv) errors.

Figure 3. MP equations (i)–(iv) errors of the ZNNFRDMP, ZNNSVDMP and ZNNP models solutions in NEs in Sections 3.1–3.5.

3.3. Experiment 3

Assume the following 4×4 input matrix of rank 1 with $p = 1$:

$$A(t) = [5 - \eta(\pi t), 3 + \theta(\pi t), -4 - \eta(\pi t), 1 + 3\theta(\pi t)] \odot \mathbf{1}_{4\times 4}.$$

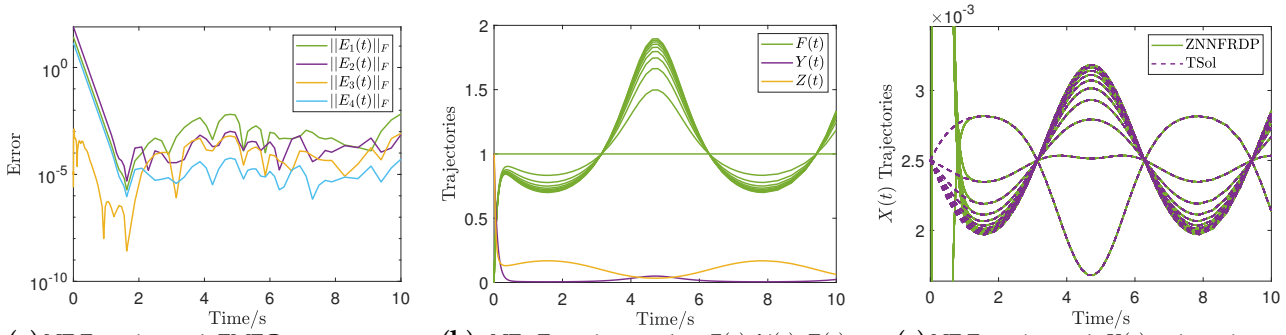
Presuming that $C(t) = A(:, p)(t)$, the findings of the ZNNFRDP model are displayed in Figures 2g–i and 3e,f.

3.4. Experiment 4

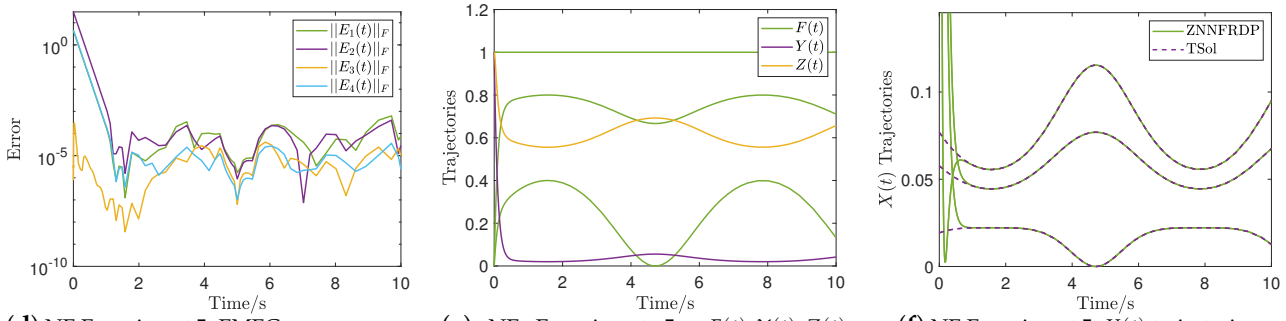
Assume the following $m \times n$ input matrix of rank 1 with $p = 1$, $m = 20$ and $n = 10$:

$$A(t) = [2 + \theta(t), 2 + 1/2\theta(t), 2 + 1/3\theta(t), \ldots, 2 + 1/n\theta(t)] \odot \mathbf{1}_{m \times n}.$$

Presuming that $C(t) = A(:, p)(t)$, the findings of the ZNNFRDP model are displayed in Figures 3g,h and 4a–c.



(a) NE Experiment 4: EMEG convergence. **(b)** NE Experiment 4: $F(t), Y(t), Z(t)$ trajectories. **(c)** NE Experiment 4: $X(t)$ trajectories.



(d) NE Experiment 5: EMEG convergence. **(e)** NE Experiment 5: $F(t), Y(t), Z(t)$ trajectories. **(f)** NE Experiment 5: $X(t)$ trajectories.

Figure 4. EMEG convergence and solutions' trajectories of the ZNNFRDMP model in NEs in Sections 3.4–3.5.

3.5. Experiment 5

Assume the following 2×3 input matrix of rank 2 with $p = [1, 2]$:

$$A(t) = \begin{bmatrix} 4 & 3 & 1 \\ 4 & 3 + 10^{-20} & 1 \end{bmatrix} + \theta(t) \odot \mathbf{1}_{2 \times 3}.$$

It is worth noting that the matrix A is close to a loss of rank, i.e., its singular values are 7.2111 and $4.6962 \cdot 10^{-18}$. Such an experiment will show how the ZNNFRDP model converges in the case of ill-conditioned matrices. Presuming that $C(t) = A(:, p)(t)$, the findings of the ZNNFRDP model are displayed in Figures 3i,j and 4d–f.

4. Discussion

This section first presents the performance of the ZNNFRDP model for computing time-varying pseudoinverses before comparing it to the performance of the ZNNP and ZNNSVDP models.

The performance of the ZNNFRDP model for computing time-varying pseudoinverses is investigated through NEs in Sections 3.1–3.5, with the results shown in Figures 2 and 4. Particularly, the EMEG convergence in NEs in Sections 3.1–3.5 is measured by the Frobenius norm in Figure 2a,d,g and Figure 4a,d, respectively. These figures demonstrate how the EMEG's $E_i(t), i = 1, 2, 3$ begin at $t = 0$ with a high error value above 10^0 and converge before $t = 2$ in a low error value in the range of $[10^{-13}, 10^{-2}]$. As opposed to this, the EMEG's

$E_4(t)$ always begins at $t = 0$ with a low error value under 10^{-3} and converges before $t = 2$ in an even lower error value in the range of $[10^{-13}, 10^{-4}]$. That is, the EMEG starts from a non-optimal initial price and converges to the zero matrix finally. The trajectories of $F(t), Y(t)$ and $Z(t)$ in NEs in Sections 3.1–3.5 are shown in Figure 2b,e,h and Figure 4b,e, respectively, while the trajectories of $X(t)$ are shown in Figure 2c,f,i and Figure 4c,f, respectively. All of these images show that while the trajectories of $X(t)$ match those of the TSol, a certain pattern appears in the trajectories of $F(t), Y(t), Z(t)$ and $X(t)$ before $t = 2$.

In general, employing the FRD to compute the pseudoinverse as suggested in (2) will increase rounding errors and result in a loss of precision when using conventional computational methods. However, our study uses the ZNN method, which is considerably different from the conventional computational methods. In fact, for example, we are not calculating the inverse of any matrix by using conventional textbook methods. Numerous papers and books (written in the last two decades) on the ZNN method demonstrate how effective the ZNN design is in attempting to zero out rounding errors by defining an appropriate EMEG (see (6)). It is important to mention that a rounding error is the difference between a rounded-off numerical value and the actual value. Considering that the EMEG's actual solution is the zero matrix, any EMEG's element that is not zero during the computations is regarded as a rounding error. As a result, the actual task of the ZNN method is to zero the rounding errors.

It is important to note that the steady-state errors in Figures 2 and 3 are of order 10^{-5} in the majority of situations because double precision arithmetic ($eps = 2.22 \cdot 10^{-16}$) is used by default in MATLAB. Keep in mind that the symbol eps stands for the machine epsilon, an upper bound on the relative approximation error brought on by rounding in floating point arithmetic. Therefore, in order to achieve lower steady-state errors, a higher error tolerance in `ode15s` must be chosen. Furthermore, the time of convergence, which in the NEs is close to $t = 2$, and the value of λ are correlated. To decrease the time of convergence, a larger value of λ must be chosen [24]. Therefore, the major factor used to rate the ZNN models' overall performance is the their convergence performance under the same value of λ.

The performance of the ZNNFRDP, ZNNP and ZNNSVDP models is compared by measuring the errors caused in the Moore-Penrose equations (i)–(iv) from (1) through NEs in Sections 3.1–3.5, with the results shown in Figure 3. Particularly, the performance of the ZNNFRDP and ZNNP models is compared through NEs in Sections 3.1 and 3.2. The results of the ZNNFRDP and ZNNP models are shown in Figure 3a,b, where we observe that in NE in Section 3.1, the errors caused in the Moore-Penrose equations (i)–(iii) and (ii)–(iv), respectively, are shown in Figure 3a,b. These figures demonstrate how these errors begin at $t = 0$ with a high value above 10^0 and converge before $t = 2$ in a low error value in the range of $[10^{-6}, 10^{-3}]$. The overall performance of the ZNNFRDP model in NE in Section 3.1 is almost identical to that of the ZNNP. In NE in Section 3.2, the errors caused in the Moore-Penrose equations (i)–(iii) and (ii)–(iv), respectively, are shown in Figure 3c,d. These figures demonstrate how these errors begin at $t = 0$ with a high value above 10^0 and converge before $t = 2$ in a low error value in the range of $[10^{-10}, 10^{-3}]$. The convergence performance of the ZNNFRDP model in NE in Section 3.2 is slightly better than that of the ZNNP, while their overall performances are almost identical. As a result, the ZNNFRDP model performed slightly better than the ZNNP in NEs in Sections 3.1 and 3.2.

Additionally, the performance of the ZNNFRDP and ZNNSVDP models is compared through NEs in Sections 3.3–3.5. It is important to note that NE in Section 3.4 compares the performances of these models in a high-dimensional input matrix with dimensions 20×10, whereas NE in Section 3.5 compares their performance in an ill-conditioned input matrix. In NE in Section 3.3, the errors caused in the Moore-Penrose equations (i)–(iii) and (ii)–(iv), respectively, are shown in Figure 3e,f. These figures demonstrate how the errors caused in the Moore-Penrose equations (i), (ii) and (iv) begin at $t = 0$ with a high value above 10^0 and converge before $t = 2$ in a low error value in the range of $[10^{-7}, 10^{-3}]$, whereas the error caused in the Moore-Penrose Equation (i) begins at $t = 0$ with a lower

value and converges before $t = 2$ in a lower error value in the range of $[10^{-11}, 10^{-5}]$. The convergence performance of the ZNNFRDP model in NE in Section 3.3 is slightly better than that of the ZNNSVDP, while their overall performances are close. In NEs in Sections 3.4 and 3.5, the errors caused in the Moore-Penrose equations (i)–(iii), respectively, are shown in Figure 3a,b, while the errors caused in the Moore-Penrose equations (ii)–(iv), respectively, are shown in Figure 3a,b. These figures demonstrate how the errors caused in the Moore-Penrose equations (ii) and (iv) begin at $t = 0$ with a high value above 10^0 and converge before $t = 2$ in a low error value in the range of $[10^{-7}, 10^{-3}]$, where the convergence performance of the ZNNFRDP model is slightly worse than that of the ZNNSVDP, while the overall performance of the ZNNFRDP model is slightly better than that of the ZNNSVDP. The error caused in the Moore-Penrose Equation (i) begin at $t = 0$ with a lower value and converge before $t = 2$ in a lower error value in the range of $[10^{-5}, 10^{-2}]$. The convergence and the overall performance of the ZNNFRDP models are almost identical. The error caused in the Moore-Penrose Equation (iii) begin at $t = 0$ with a lower value and converge before $t = 2$ in a lower error value in the range of $[10^{-12}, 10^{-5}]$. The convergence and the overall performance of the ZNNFRDP model are much better than that of the ZNNSVDP. As a result, the ZNNFRDP model performed better than the ZNNSVDP in NEs in Sections 3.3–3.5.

Finally, it was demonstrated in Section 2 that the ZNNFRDP model has an overall computational complexity of $\mathcal{O}((w + 2h + mn)^3)$, whereas [11] claims that the ZNNP model has an overall computational complexity of $\mathcal{O}((mn)^3)$ and the ZNNSVDP model has an overall computational complexity of $\mathcal{O}((m^2 + n^2 + r + mn)^3)$. Therefore, the ZNNFRDP model has a lower overall computational complexity than the ZNNSVDP model, which is an advantage, but a higher overall computational complexity than the ZNNP model, which is a disadvantage.

To summarize, the following can be deduced from this section's NEs:

- The convergence of the ZNNFRDP model starts from a non-optimal initial price and converge in a small period of time to the zero matrix.
- The errors caused in the Moore-Penrose equations (i)–(iv) and the $F(t), Y(t), Z(t)$ and $X(t)$ solutions trajectories act consistently due to the EMEGs' proclivity toward convergence.
- The EMEGs will converge more quickly when the price of the design parameter λ is greater.
- The ZNNFRDP model performs as well as, if not better than, the ZNNP and ZNNSVDP models.
- In essence, the time-varying pseudoinverses are calculated with exceptional and efficient performance by the ZNNFRDP model.

5. Conclusions

In this paper, the issue of calculating the time-varying pseudoinverse based on FRD using the ZNN method was examined. Therefore, a novel ZNN model called ZNNFRDP that uses the FRD to compute the time-varying pseudoinverse was proposed and explored in this research. Five NEs show that the ZNNFRDP model's EMEG has a tendency to converge exponentially to the zero matrix after beginning from a non-optimal initial price. Additionally, they show that the ZNNFRDP model performs as well as, if not better than, other well-performing ZNN models in the computation of the time-varying pseudoinverse. Thus, theoretical analysis and numerical results both supported the effectiveness of the proposed model.

The list below includes some possible research topics.

1. To hasten the convergence of the ZNN dynamics, nonlinear activation functions [37,38] may be used.
2. It may be possible to use ZNN designs with terminal convergence [39,40] to produce new improved ZNNFRDP models.

Author Contributions: All authors (H.A., H.J., M.K., R.A., T.E.S., S.D.M. and V.N.K.) contributed equally. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Research Deanship of Hail University-KSA Project Number (RG -21 127).

Data Availability Statement: Not applicable.

Acknowledgments: Authors acknowledge the Research Deanship of Hail University-KSA for the administrative, financial, and technical support.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zhang, S.; Dong, Y.; Ouyang, Y.; Yin, Z.; Peng, K. Adaptive neural control for robotic manipulators with output constraints and uncertainties. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *29*, 5554–5564. [CrossRef] [PubMed]
2. Mourtas, S.; Katsikis, V.; Kasimis, C. Feedback control systems stabilization using a bio-inspired neural network. *EAI Endorsed Trans. AI Robot.* **2022**, *1*, 1–13. [CrossRef]
3. Yuan, Y.; Wang, Z.; Guo, L. Event-triggered strategy design for discrete-time nonlinear quadratic games with disturbance compensations: The noncooperative case. *IEEE Trans. Syst. Man Cybern. Syst.* **2018**, *48*, 1885–1896. [CrossRef]
4. Yang, X.; He, H. Self-learning robust optimal control for continuous-time nonlinear systems with mismatched disturbances. *Neural Netw.* **2018**, *99*, 19–30. [CrossRef] [PubMed]
5. Mourtas, S.D. A weights direct determination neuronet for time-series with applications in the industrial indices of the federal reserve bank of St. Louis. *J. Forecast.* **2022**, *14*, 1512–1524. [CrossRef]
6. Li, S.; He, J.; Li, Y.; Rafique, M.U. Distributed recurrent neural networks for cooperative control of manipulators: A game-theoretic perspective. *IEEE Trans. Neural Netw. Learn. Syst.* **2017**, *28*, 415–426. [CrossRef]
7. Ben-Israel, A.; Greville, T.N.E. *Generalized Inverses: Theory and Applications*, 2nd ed.; CMS Books in Mathematics; Springer: New York, NY, USA, 2003. [CrossRef]
8. Wang, G.; Wei, Y.; Qiao, S.; Lin, P.; Chen, Y. *Generalized Inverses: Theory and Computations*; Springer: Singapore, 2018; Volume 53.
9. Tan, N.; Yu, P.; Ni, F. New varying-parameter recursive neural networks for model-free kinematic control of redundant manipulators with limited measurements. *IEEE Trans. Instrum. Meas.* **2022**, *71*, 3161713. [CrossRef]
10. Zhong, N.; Huang, Q.; Yang, S.; Ouyang, F.; Zhang, Z. A varying-parameter recurrent neural network combined with penalty function for solving constrained multi-criteria optimization scheme for redundant robot manipulators. *IEEE Access* **2021**, *9*, 50810–50818. [CrossRef]
11. Kornilova, M.; Kovalnogov, V.; Fedorov, R.; Zamaleev, M.; Katsikis, V.N.; Mourtas, S.D.; Simos, T.E. Zeroing neural network for pseudoinversion of an arbitrary time-varying matrix based on singular value decomposition. *Mathematics* **2022**, *10*, 1208. [CrossRef]
12. Aleskerov, F.; Ersel, H.; Piontkovski, D. *Linear Algebra for Economists*; Springer Texts in Business and Economics; Springer: Berlin/Heidelberg, Germany, 2011. [CrossRef]
13. Gupta, A.K. *Numerical Methods Using MATLAB*; MATLAB solutions series; Apress: Berkeley, CA, USA, 2014.
14. Katsikis, V.N.; Pappas, D.; Petralias, A. An improved method for the computation of the Moore–Penrose inverse matrix. *Appl. Math. Comput.* **2011**, *217*, 9828–9834. [CrossRef]
15. Stanimirović, P.S.; Roy, F.; Gupta, D.K.; Srivastava, S. Computing the Moore-Penrose inverse using its error bounds. *Appl. Math. Comput.* **2020**, *371*, 124957. [CrossRef]
16. Leithead, W.E.; Zhang, Y. $O(N^2)$-operation approximation of covariance matrix inverse in Gaussian process regression based on quasi-Newton BFGS method. *Commun. Stat. Simul. Comput.* **2007**, *36*, 367–380. [CrossRef]
17. Zhang, Z.; Lu, Y.; Zheng, L.; Li, S.; Yu, Z.; Li, Y. A new varying-parameter convergent-differential neural-network for solving time-varying convex QP problem constrained by linear-equality. *IEEE Trans. Autom. Control* **2018**, *63*, 4110–4125. [CrossRef]
18. Simos, T.E.; Katsikis, V.N.; Mourtas, S.D.; Stanimirović, P.S.; Gerontitis, D. A higher-order zeroing neural network for pseudoinversion of an arbitrary time-varying matrix with applications to mobile object localization. *Inf. Sci.* **2022**, *600*, 226–238. [CrossRef]
19. Zhang, Z.; Zheng, L.; Yu, J.; Li, Y.; Yu, Z. Three recurrent neural networks and three numerical methods for solving a repetitive motion planning scheme of redundant robot manipulators. *IEEE/ASME Trans. Mechatron.* **2017**, *22*, 1423–1434. [CrossRef]
20. Katsikis, V.N.; Mourtas, S.D.; Stanimirović, P.S.; Li, S.; Cao, X. Time-varying mean-variance portfolio selection problem solving via LVI-PDNN. *Comput. Oper. Res.* **2022**, *138*, 105582. [CrossRef]
21. Bai, L.; Zheng, K.; Wang, Z.; Liu, J. Service provider portfolio selection for project management using a BP neural network. *Ann. Oper. Res.* **2022**, *308*, 41–62. [CrossRef]
22. Zhang, Y.; Guo, D.; Li, Z. Common nature of learning between back-propagation and Hopfield-type neural networks for generalized matrix inversion with simplified models. *IEEE Trans. Neural Netw. Learn. Syst.* **2013**, *24*, 579–592. [CrossRef]
23. Lv, X.; Xiao, L.; Tan, Z.; Yang, Z.; Yuan, J. Improved gradient neural networks for solving Moore–Penrose inverse of full-rank matrix. *Neural Process. Lett.* **2019**, *50*, 1993–2005. [CrossRef]

24. Zhang, Y.; Ge, S.S. Design and analysis of a general recurrent neural network model for time-varying matrix inversion. *IEEE Trans. Neural Netw.* **2005**, *16*, 1477–1490. [CrossRef]
25. Chai, Y.; Li, H.; Qiao, D.; Qin, S.; Feng, J. A neural network for Moore-Penrose inverse of time-varying complex-valued matrices. *Int. J. Comput. Intell. Syst.* **2020**, *13*, 663–671. [CrossRef]
26. Sun, Z.; Li, F.; Jin, L.; Shi, T.; Liu, K. Noise-tolerant neural algorithm for online solving time-varying full-rank matrix Moore-Penrose inverse problems: A control-theoretic approach. *Neurocomputing* **2020**, *413*, 158–172. [CrossRef]
27. Wu, W.; Zheng, B. Improved recurrent neural networks for solving Moore-Penrose inverse of real-time full-rank matrix. *Neurocomputing* **2020**, *418*, 221–231. [CrossRef]
28. Zhang, Y.; Yang, Y.; Tan, N.; Cai, B. Zhang neural network solving for time-varying full-rank matrix Moore-Penrose inverse. *Computing* **2011**, *92*, 97–121. [CrossRef]
29. Qiao, S.; Wang, X.Z.; Wei, Y. Two finite-time convergent Zhang neural network models for time-varying complex matrix Drazin inverse. *Linear Algebra Its Appl.* **2018**, *542*, 101–117. [CrossRef]
30. Qiao, S.; Wei, Y.; Zhang, X. Computing time-varying ML-weighted pseudoinverse by the Zhang neural networks. *Numer. Funct. Anal. Optim.* **2020**, *41*, 1672–1693. [CrossRef]
31. Wang, X.; Stanimirovic, P.S.; Wei, Y. Complex ZFs for computing time-varying complex outer inverses. *Neurocomputing* **2018**, *275*, 983–1001. [CrossRef]
32. Zhang, H.; Wan, L. Zeroing neural network methods for solving the Yang-Baxter-like matrix equation. *Neurocomputing* **2020**, *383*, 409–418. [CrossRef]
33. Dai, J.; Yang, X.; Xiao, L.; Jia, L.; Liu, X.; Wang, Y. Design and analysis of a self-adaptive zeroing neural network for solving time-varying quadratic programming. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, 1–10. . [CrossRef]
34. Dai, J.; Tan, P.; Yang, X.; Xiao, L.; Jia, L.; He, Y. A fuzzy adaptive zeroing neural network with superior finite-time convergence for solving time-variant linear matrix equations. *Knowl.-Based Syst.* **2022**, *242*, 108405. [CrossRef]
35. Xiao, L.; Tan, H.; Dai, J.; Jia, L.; Tang, W. High-order error function designs to compute time-varying linear matrix equations. *Inf. Sci.* **2021**, *576*, 173–186. [CrossRef]
36. Graham, A. *Kronecker Products and Matrix Calculus with Applications*; Courier Dover Publications, Mineola, NY, USA 2018.
37. Liao, B.; Zhang, Y. From different ZFs to different ZNN models accelerated via Li activation functions to finite-time convergence for time-varying matrix pseudoinversion. *Neurocomputing* **2014**, *133*, 512–522. [CrossRef]
38. Wang, X.Z.; Ma, H.; Stanimirović, P.S. Nonlinearly activated recurrent neural network for computing the Drazin inverse. *Neural Process. Lett.* **2017**, *46*, 195. [CrossRef]
39. Xiao, L. A finite-time convergent neural dynamics for online solution of time-varying linear complex matrix equation. *Neurocomputing* **2015**, *167*, 254–259. [CrossRef]
40. Xiao, L. A nonlinearly-activated neurodynamic model and its finite-time solution to equality-constrained quadratic optimization with nonstationary coefficients. *Appl. Soft Comput.* **2016**, *40*, 252–259. [CrossRef]