Recurrent Neural Network Approach to Computation of Generalized Inverses

Predrag S. Stanimirović

May 31, 2016

Introduction

- The problem of generalized inverses computation is closely related with the following Penrose equations:
  (1) $AXA = A$ (2) $XAX = X$
  (3) $(AX)^* = AX$ (4) $(XA)^* = XA$.

Introduction

- The problem of generalized inverses computation is closely related with the following Penrose equations:
  (1) $AXA = A$ (2) $XAX = X$
  (3) $(AX)^* = AX$ (4) $(XA)^* = XA$.
- The Drazin inverse $X = A^{\mathrm{D}}$ is the unique matrix which fulfills the matrix equation (2): $XAX = X$ in conjunction with
  (1^k) $A^{k+1}X = A^k,\ k \geq \mathrm{ind}(A),$ (5) $AX = XA$.

Introduction

- The problem of generalized inverses computation is closely related with the following Penrose equations:
  (1) $AXA = A$ (2) $XAX = X$
  (3) $(AX)^* = AX$ (4) $(XA)^* = XA$.
- The Drazin inverse $X = A^{\mathrm{D}}$ is the unique matrix which fulfills the matrix equation (2): $XAX = X$ in conjunction with
  (1^k) $A^{k+1}X = A^k,\ k \geq \mathrm{ind}(A)$, (5) $AX = XA$.
- For a subset $\mathcal{S} \subseteq \{1, 2, 3, 4\}$, the set of all matrices obeying the conditions contained in $\mathcal{S}$ is denoted by $A\{\mathcal{S}\}$.

Introduction

- The problem of generalized inverses computation is closely related with the following Penrose equations:
  (1) $AXA = A$ (2) $XAX = X$
  (3) $(AX)^* = AX$ (4) $(XA)^* = XA$.
- The Drazin inverse $X = A^{\mathrm{D}}$ is the unique matrix which fulfills the matrix equation (2): $XAX = X$ in conjunction with
  (1^k) $A^{k+1}X = A^k,\ k \geq \mathrm{ind}(A)$, (5) $AX = XA$.
- For a subset $\mathcal{S} \subseteq \{1, 2, 3, 4\}$, the set of all matrices obeying the conditions contained in $\mathcal{S}$ is denoted by $A\{\mathcal{S}\}$.
  - For any matrix A there exists a single element in the set $A\{1, 2, 3, 4\}$, called the Moore-Penrose inverse of A and denoted by $A^\dagger$.

Introduction

- The outer inverse of $A \in \mathbb{C}_r^{m\times n}$ with prescribed range T and null space S, denoted by $A^{(2)}_{T,S}$, satisfies the matrix equation (2): $XAX = X$ and two additional properties: $\mathcal{R}(X) = T$ and $\mathcal{N}(X) = S$.

Introduction

- The outer inverse of $A \in \mathbb{C}_r^{m\times n}$ with prescribed range T and null space S, denoted by $A^{(2)}_{T,S}$, satisfies the matrix equation (2): $XAX = X$ and two additional properties: $\mathcal{R}(X) = T$ and $\mathcal{N}(X) = S$.
- The Moore-Penrose inverse $A^\dagger$ and the weighted Moore-Penrose inverse $A^\dagger_{M,N}$, the Drazin inverse A^D and the group inverse $A^\#$ can be derived by means of appropriate choices of T and S:
$A^\dagger = A^{(2)}_{\mathcal{R}(A^*),\mathcal{N}(A^*)}$, $A^\dagger_{M,N} = A^{(2)}_{\mathcal{R}(A^\sharp),\mathcal{N}(A^\sharp)}$, $A^\sharp = N^{-1}A^*M$
$A^{\mathrm{D}} = A^{(2)}_{\mathcal{R}(A^k),\mathcal{N}(A^k)}$, $k \geq \mathrm{ind}(A)$, $A^\# = A^{(2)}_{\mathcal{R}(A),\mathcal{N}(A)}$.

Introduction

- The main efforts in the generalized inverse computation can be divided into two main types: numerical algorithms and continuous-time neural network algorithms.

Introduction

- The numerical algorithms can be divided in two categories: direct and iterative methods.

Introduction

- The numerical algorithms can be divided in two categories: direct and iterative methods.
- The direct methods can be divided as:

Introduction

- The numerical algorithms can be divided in two categories: direct and iterative methods.
- The direct methods can be divided as:
  - The singular value decomposition (SVD) algorithm is the most known between the direct methods.

Introduction

- The numerical algorithms can be divided in two categories: direct and iterative methods.
- The direct methods can be divided as:
  - The singular value decomposition (SVD) algorithm is the most known between the direct methods.
  - Also, other types of matrix factorizations has been exploited in computation of generalized inverses, such as the QR decomposition, LU factorization.

Introduction

- The numerical algorithms can be divided in two categories: direct and iterative methods.
- The direct methods can be divided as:
  - The singular value decomposition (SVD) algorithm is the most known between the direct methods.
  - Also, other types of matrix factorizations has been exploited in computation of generalized inverses, such as the QR decomposition, LU factorization.
  - Methods based on the application of the Gauss-Jordan elimination process to an appropriate augmented matrix were also investigated.

Introduction

- The numerical algorithms can be divided in two categories: direct and iterative methods.
- The direct methods can be divided as:
  - The singular value decomposition (SVD) algorithm is the most known between the direct methods.
  - Also, other types of matrix factorizations has been exploited in computation of generalized inverses, such as the QR decomposition, LU factorization.
  - Methods based on the application of the Gauss-Jordan elimination process to an appropriate augmented matrix were also investigated.
  - The SVD algorithm is more accurate and is thus the most commonly used method, but it requires a large amount of computational resources.

Introduction

- The iterative methods, such as the orthogonal projection algorithms, the Newton iterative algorithm, and the higher-order convergent iterative methods as well as the methods based on optimization are more suitable for implementation.

Introduction

- The iterative methods, such as the orthogonal projection algorithms, the Newton iterative algorithm, and the higher-order convergent iterative methods as well as the methods based on optimization are more suitable for implementation.
- All iterative methods, in general, require initial conditions which are ultimate, rigorous and sometimes cannot be fulfilled easily.

Introduction

- The continuous-time neural learning algorithms have emerged as parallel distributed computational models for real-time applications.

Introduction

- The continuous-time neural learning algorithms have emerged as parallel distributed computational models for real-time applications.
  - In many real-time systems, real-time solutions of pseudoinverse matrices are usually imperative. An example of such applications is application in robotics.

Introduction

- The continuous-time neural learning algorithms have emerged as parallel distributed computational models for real-time applications.
  - In many real-time systems, real-time solutions of pseudoinverse matrices are usually imperative. An example of such applications is application in robotics.
- The starting point of our investigations in developing recurrent neural network (RNNs) were previously developed neural network models for the matrix inversion as well as for the pseudoinversion of full-rank and rank-deficient rectangular matrices.

GNN in time-invariant case

Neural network approach restricted by the spectrum

Our intention was to develop and investigate the RNN approach in computation of the Drazin and group inverse and later in computation of $\{2\}$-inverses with prescribed range and null space, in both the time invariant and the time-varying case.

GNN in time-invariant case

Neural network approach restricted by the spectrum

Our intention was to develop and investigate the RNN approach in computation of the Drazin and group inverse and later in computation of $\{2\}$-inverses with prescribed range and null space, in both the time invariant and the time-varying case.

- Two GNNs for computing $\{2\}$-inverses with prescribed range and null space are defined in
  [I. Živković, P.S. Stanimirović, Y. Wei, *Recurrent Neural Network for Computing Outer Inverses*, Neural Computation **28** (2016), 970–998.]

Time-invariant case

Neural network approach restricted by the spectrum

- Step 1. Exploit a matrix equation corresponding to a generalized inverse or a class of generalized inverses.

Time-invariant case

Neural network approach restricted by the spectrum

- Step 1. Exploit a matrix equation corresponding to a generalized inverse or a class of generalized inverses.

Lemma 1

Let $A \in \mathbb{C}_r^{m\times n}$ be given and $G \in \mathbb{C}_s^{n\times m}$ be arbitrarily chosen matrix satisfying $0 < s \leq r$. Assume that $X := A^{(2)}_{\mathcal{R}(G),\mathcal{N}(G)}$ exists. Then the matrix equations

$$GAX = G, \quad XAG = G$$

are satisfied.

GNN in time-invariant case

Neural network approach restricted by the spectrum

- The equation $GAX = G$ can be rewritten as the dynamic matrix equation

$$GAV_G(t) - G = 0,$$

where $V_G \in \mathbb{R}^{n\times m}$ denotes the unknown matrix to be solved and which corresponds to the outer inverse with prescribed range and null space $X := A^{(2)}_{\mathcal{R}(G),\mathcal{N}(G)}$.

GNN in time-invariant case

Neural network approach restricted by the spectrum

Our intention is to solve $GAV_G(t) - G = 0$ for the unknown matrix V_G using dynamic-system approach.

GNN in time-invariant case

Neural network approach restricted by the spectrum

Our intention is to solve $GAV_G(t) - G = 0$ for the unknown matrix V_G using dynamic-system approach.

- Step 2. A common approach in that purpose is to define a scalar-valued norm based error function. In our case,

$$E(t) = \frac{\|GAV_G(t) - G\|_F^2}{2}.$$

GNN in time-invariant case

Neural network approach restricted by the spectrum

- Step 3. A computational scheme for computing the minimum point $\overline{V_G}$ could be defined along the gradient descent direction of $E(t)$.

GNN in time-invariant case

Neural network approach restricted by the spectrum

- Step 3. A computational scheme for computing the minimum point $\overline{V_G}$ could be defined along the gradient descent direction of $E(t)$.
  - The derivative of $E(t)$ with respect to $V_G \in \mathbb{R}^{n \times m}$ could be derived applying the principles of the matrix calculus:

$$\frac{\partial E(t)}{\partial V_G} = (GA)^{\mathrm{T}} (GAV_G(t) - G).$$

GNN in time-invariant case

Neural network approach restricted by the spectrum

- Step 3. A computational scheme for computing the minimum point $\overline{V_G}$ could be defined along the gradient descent direction of $E(t)$.
- The derivative of $E(t)$ with respect to $V_G \in \mathbb{R}^{n \times m}$ could be derived applying the principles of the matrix calculus:

$$\frac{\partial E(t)}{\partial V_G} = (GA)^{\mathrm{T}} \left(GAV_G(t) - G \right).$$

The constant term $(GA)^{\mathrm{T}}$ in the last identity can be omitted without loss of generality.

GNN in time-invariant case

Neural network approach restricted by the spectrum

- Step 4. Define the dynamic equation of the underlying recurrent neural network (RNN) using

$$\frac{dV_G(t)}{dt} = -\beta \frac{\partial E(t)}{\partial V_G}.$$

GNN in time-invariant case

Neural network approach restricted by the spectrum

- Step 4. Define the dynamic equation of the underlying recurrent neural network (RNN) using

$$\frac{dV_G(t)}{dt} = -\beta \frac{\partial E(t)}{\partial V_G}.$$

In this way, the dynamic equation of the initiated recurrent neural network (called GNNATS2-I) is given in the form

$$\begin{cases} \frac{dV_G(t)}{dt} = -\beta \left(GAV_G(t) - G \right), V(0) = 0, \ \ if \ \ m \geq n, \\ \frac{dV_G(t)}{dt} = -\beta \left(V_G(t)AG - G \right), V(0) = 0, \ \ if \ \ m < n. \end{cases}$$

GNN in time-invariant case

Neural network approach restricted by the spectrum

Theorem 1

Let $A \in \mathbb{R}_r^{m\times n}$ be a given matrix, $G \in \mathbb{R}_s^{n\times m}$ be arbitrary matrix satisfying $0 < s \leq r$, and $\sigma(GA) = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ be the spectrum of GA. Suppose that the condition

$$\mathrm{Re}\left(\sigma(GA)\right) \geq 0 \equiv \mathrm{Re}(\lambda_j) \geq 0,\ j = 1, 2, \ldots, n$$

is satisfied.
Then the model GNNATS2-I gives the result

$$\overline{V_G} = \lim_{t\to\infty} V_G(t) = A^{(2)}_{\mathcal{R}(G),\mathcal{N}(G)}.$$

GNN in time-invariant case

Neural network approach dependent on the restriction on spectrum

- According to the last theorem, the application of the dynamic equation GNNATS2-I is conditioned by the properties of the spectrum of the matrix GA or AG:

$$\begin{cases} \mathrm{Re}\,(\sigma(GA)) \geq 0 \equiv \sigma(GA) \subset \{z : Re(z) \geq 0\}, \ m \geq n, \\ \mathrm{Re}\,(\sigma(AG)) \geq 0 \equiv \sigma(AG) \subset \{z : Re(z) \geq 0\}, \ m < n. \end{cases}$$

GNN in time-invariant case

Neural network approach dependent on the restriction on spectrum

- According to the last theorem, the application of the dynamic equation GNNATS2-I is conditioned by the properties of the spectrum of the matrix GA or AG:

$$\begin{cases} \mathrm{Re}\,(\sigma(GA)) \geq 0 \equiv \sigma(GA) \subset \{z : Re(z) \geq 0\}, & m \geq n, \\ \mathrm{Re}\,(\sigma(AG)) \geq 0 \equiv \sigma(AG) \subset \{z : Re(z) \geq 0\}, & m < n. \end{cases}$$

- More precisely, the first GNNATS2-I model fails in the case when $\mathrm{Re}\,(\sigma(GA))$ contains negative values.

GNN in time-invariant case

Neural network approach restricted by the spectrum

- The neural network defined by GNNATS2-I used in our implementation is composed from a number of independent subnetworks.

GNN in time-invariant case

Neural network approach restricted by the spectrum

- The neural network defined by GNNATS2-I used in our implementation is composed from a number of independent subnetworks.
  Let us denote by $v_j(t)$ (resp. g_j) the jth column vector of $V_G(t)$ (resp. G), for $j = 1, 2, \ldots, m$. The dynamics of the jth subnetwork of GNNATS2-I is in the general form:

$$\frac{dv_j(t)}{dt} = -\beta \left(GAv_j(t) - g_j \right).$$

GNN in time-invariant case

Neural network approach restricted by the spectrum

- The neural network defined by GNNATS2-I used in our implementation is composed from a number of independent subnetworks.
  Let us denote by $v_j(t)$ (resp. g_j) the jth column vector of $V_G(t)$ (resp. G), for $j = 1, 2, \ldots, m$. The dynamics of the jth subnetwork of GNNATS2-I is in the general form:

$$\frac{dv_j(t)}{dt} = -\beta \left(GAv_j(t) - g_j \right).$$

  Each subnetwork exploits the same connection weight matrix $W = -\beta GA$ and $\beta g_j = \{\beta g_{1j}, \beta g_{2j}, \ldots, \beta g_{nj}\}$ is the biasing threshold vector for the jth subnetwork.

GNN in time-invariant case

Neural network approach dependent on the restriction on spectrum

- Elements v_{ij} of unknown matrix $V_G(t)$ are computed using the system of differential equations

$$\dot{v}_{ij} = \frac{dv_{ij}}{dt} = \sum_{k=1}^{n} w_{ik} v_{kj} + \beta g_{ji},$$
$$i = 1, 2, \ldots, n; j = 1, 2, \ldots, m,$$

where w_{ij} are elements of $W = -\beta GA$ and v_{ij} are elements of $V_G(t)$.

GNN in time-invariant case

Neural network approach dependent on the restriction on spectrum

- Elements v_{ij} of unknown matrix $V_G(t)$ are computed using the system of differential equations

$$\dot{v}_{ij} = \frac{dv_{ij}}{dt} = \sum_{k=1}^{n} w_{ik} v_{kj} + \beta g_{ji},$$
$$i = 1, 2, \ldots, n; j = 1, 2, \ldots, m,$$

  where w_{ij} are elements of $W = -\beta GA$ and v_{ij} are elements of $V_G(t)$.
- It is important to mention that the column vector $v_j = \{v_{1j}, v_{2j}, \ldots, v_{nj}\}$ is output in the jth subnetwork.

Time-invariant case

Neural network approach without restriction on the spectrum

An approach to resolve the restriction $\mathrm{Re}\,(\sigma(GA)) \geq 0$ was proposed in
[I. Živković, P.S. Stanimirović, Y. Wei, *Recurrent Neural Network for Computing Outer Inverses*, Neural Computation **28** (2016), 970–998.]

Time-invariant case

Neural network approach without restriction on the spectrum

An approach to resolve the restriction $\mathrm{Re}\,(\sigma(GA)) \geq 0$ was proposed in
[I. Živković, P.S. Stanimirović, Y. Wei, *Recurrent Neural Network for Computing Outer Inverses*, Neural Computation **28** (2016), 970–998.]

- The dynamical equation and the initiated GNN are denoted by GNNATS2-II and they are based on the replacement of G by $G_0 = G(GAG)^T G$ in GNNATS2-I:

$$\begin{cases} \frac{dV_{G_0}(t)}{dt} = -\beta\,(G_0 A V_{G_0}(t) - G_0)\,, \;\; V(0) = 0, \textit{if} \;\; m \geq n, \\ \frac{dV_{G_0}(t)}{dt} = -\beta\,(V_{G_0}(t) A G_0 - G_0)\,, \;\; V(0) = 0, \textit{if} \;\; m < n. \end{cases}$$

GNN in time-invariant case

Neural network approach without restriction on the spectrum

- Advantages: The GNNATS2-II model is globally convergent.

GNN in time-invariant case

Neural network approach without restriction on the spectrum

- Advantages: The GNNATS2-II model is globally convergent.
- Disadvantages: The GNNATS2-II requires additional matrix multiplications during the computation of the matrix G_0 instead of the matrix G. In the essence, it eliminates the requirements on the spectrum, at the cost of increasing the number of matrix operations.

GNN in time-invariant case

Neural network approach without restriction on the spectrum

- Advantages: The GNNATS2-II model is globally convergent.
- Disadvantages: The GNNATS2-II requires additional matrix multiplications during the computation of the matrix G_0 instead of the matrix G. In the essence, it eliminates the requirements on the spectrum, at the cost of increasing the number of matrix operations.
- Also, the numbers in G_0 grows, which could cause numerical instability.

GNN in time-invariant case

Particular cases

- In the particular case $G = A^{\mathrm{T}}$ we immediately derive known results concerning the usual inverse when matrix A is nonsingular, as well as the Moore–Penrose inverse when the matrix A is rectangular or rank-deficient.

$$\frac{dV(t)}{dt} = -\beta A^{\mathrm{T}} A V(t) + \beta A^{\mathrm{T}}, \quad V(0) = V_0.$$

GNN in time-invariant case

Particular cases

- In the particular case $G = A^{\mathrm{T}}$ we immediately derive known results concerning the usual inverse when matrix A is nonsingular, as well as the Moore–Penrose inverse when the matrix A is rectangular or rank-deficient.

$$\frac{dV(t)}{dt} = -\beta A^{\mathrm{T}} AV(t) + \beta A^{\mathrm{T}}, \quad V(0) = V_0.$$

- The choice $G = A^{\sharp} = N^{-1}A^{\mathrm{T}}M$ produces the weighted Moore-Penrose inverse $A^{\dagger}_{M,N}$.

$$\frac{dV(t)}{dt} = -\beta A^{\sharp} AV(t) + \beta A^{\sharp}$$

GNN in time-invariant case

Particular cases

The cases $G = A^{\mathrm{T}}$ and $G = A^{\sharp}$ was established initially.

GNN in time-invariant case

Particular cases

The cases $G = A^{\mathrm{T}}$ and $G = A^{\sharp}$ was established initially.
We observed that additional possibilities are available.

- In the case $G = A^l$, $l \geq \mathrm{ind}(A)$ the general GNNATS2 model could be applicable in computation of the Drazin inverse.

GNN in time-invariant case

Particular cases

The cases $G = A^{\mathrm{T}}$ and $G = A^{\sharp}$ was established initially.
We observed that additional possibilities are available.

- In the case $G = A^l$, $l \geq \mathrm{ind}(A)$ the general GNNATS2 model could be applicable in computation of the Drazin inverse.
- For a matrix A of index $\mathrm{ind}(A) = 1$ the general GNNATS2 model produces the group inverse $A^{\#}$ of A.
- The choice of an arbitrary matrix $G \in \mathbb{G}_s^{n \times m}$ satisfying $\mathrm{rank}(G) = s \leq \mathrm{rank}(A)$ is also interesting.

GNN in time-invariant case

Particular cases with restriction on the spectrum

One important case is $G = A^l,\ l \geq ind(A)$, which defines a recurrent neural network for computing the Drazin inverse of a real square matrix in real time. The choice $G = A^l$ in GNNATS2-I is called GNNAD-I.

$$\frac{dV(t)}{dt} = -\beta \left(A^{l+1} V(t) - A^l \right),\ l \geq \mathrm{ind}(A),\ V(0) = 0.$$

GNN in time-invariant case

Particular cases with restriction on the spectrum

One important case is $G = A^l,\ l \geq ind(A)$, which defines a recurrent neural network for computing the Drazin inverse of a real square matrix in real time. The choice $G = A^l$ in GNNATS2-I is called GNNAD-I.

$$\frac{dV(t)}{dt} = -\beta \left(A^{l+1}V(t) - A^l\right),\ l \geq \mathrm{ind}(A),\ V(0) = 0.$$

This case was considered in
[P.S. Stanimirović, I. Živković, Y. Wei, *Recurrent neural network for computing the Drazin inverse*, IEEE Transactions on Neural Networks and Learning Systems, **26** (2015), 2830–2843.]

GNN in time-invariant case

Particular cases with restriction on the spectrum

The main goals of that paper are:
1. The considered cases had been $GA = A^{\mathrm{T}}A$ or $GA = A^{\#}A$, where the matrices GA had been positive semidefinite. In the case $GA = A^{l}A$, the matrix A^{l+1} is not positive semidefinite, and this difficulty had to be avoided.

GNN in time-invariant case

Particular cases with restriction on the spectrum

The main goals of that paper are:
1. The considered cases had been $GA = A^{\mathrm{T}}A$ or $GA = A^{\#}A$, where the matrices GA had been positive semidefinite. In the case $GA = A^{l}A$, the matrix A^{l+1} is not positive semidefinite, and this difficulty had to be avoided.
2. An algorithm for finding the first integer l satisfying

$$\mathrm{Re}\left(\sigma(A^{l}A)\right) \geq 0 \equiv \sigma\left(A^{k+1}\right) \subset \{z : Re(z) \geq 0\} \qquad (1)$$

is defined.

GNN in time-invariant case

Particular cases with restriction on the spectrum

Also, very important particular case is $G = A^l,\ l \geq \mathrm{ind}A$ in $G_0 = G(GAG)^{\mathrm{T}}G$, i.e. in GNNATS2-II, which defines a recurrent neural network for computing the Drazin inverse of a real square matrix in real time (called GNNAD-II). This case was considered in

[P.S. Stanimirović, I. Živković, Y. Wei, *Recurrent neural network approach based on the integral representation of the Drazin inverse*, Neural Computation, **27**(10) (2015), 2107–2131.]

GNN in time-invariant case

Particular cases with restriction on the spectrum

Also, very important particular case is $G = A^l,\ l \geq \mathrm{ind}A$ in $G_0 = G(GAG)^{\mathrm{T}}G$, i.e. in GNNATS2-II, which defines a recurrent neural network for computing the Drazin inverse of a real square matrix in real time (called GNNAD-II). This case was considered in

[P.S. Stanimirović, I. Živković, Y. Wei, *Recurrent neural network approach based on the integral representation of the Drazin inverse*, Neural Computation, **27**(10) (2015), 2107–2131.]

The main goals of this paper are:

1. Avoid restrictions on the spectrum and ensure global stability.

GNN in time-invariant case

Additional dynamic state equations for outer inverses

Two additional dynamic state equations and corresponding gradient based RNNs for generating the class of outer inverses are proposed in
[P.S. Stanimirović, I. Živković, Y. Wei, *Neural network approach to computing outer inverses based on the full rank representation*, Linear Algebra Appl., **501** (2016), 344–362.]

GNN in time-invariant case

Additional dynamic state equations for outer inverses

Lemma 2

Let $A \in \mathbb{C}^{m\times n}$ be of rank r, let T be a subspace of $\mathbb{C}^n$ of dimension $s \leq r$, and let S be a subspace of $\mathbb{C}^m$ of dimension $m-s$. In addition, suppose that $G \in \mathbb{C}_s^{n\times m}$ satisfies $\mathcal{R}(G) = T$ and $\mathcal{N}(G) = S$. Assume that $G = PQ$ is a full rank factorization of G. Then $A^{(2)}_{T,S}$ exists if and only if

$$QAP \quad is\ invertible.$$

In this case

$$\begin{aligned} A^{(2)}_{T,S} &= G(AG)^{\#} = (GA)^{\#}G \\ &= P(QAP)^{-1}Q. \end{aligned}$$

GNN in time-invariant case

Additional dynamic state equations for outer inverses

- One RNN model is based on the representation

$$A^{(2)}_{\mathcal{R}(G),\mathcal{N}(G)} = G(AG)^{\#} = (GA)^{\#}G$$

of the outer inverse $A^{(2)}_{\mathcal{R}(G),\mathcal{N}(G)}$ and uses the GNNAD-I model in order to compute $(AG)^{\#}$ or $(GA)^{\#}$.

GNN in time-invariant case

Additional dynamic state equations for outer inverses

- One RNN model is based on the representation

$$A^{(2)}_{\mathcal{R}(G),\mathcal{N}(G)} = G(AG)^{\#} = (GA)^{\#}G$$

of the outer inverse $A^{(2)}_{\mathcal{R}(G),\mathcal{N}(G)}$ and uses the GNNAD-I model in order to compute $(AG)^{\#}$ or $(GA)^{\#}$.
State equation:

$$\frac{dV(t)}{dt} = -\beta \left((GA)^{l+1}V(t) - (GA)^{l}\right), l \geq \mathrm{ind}(A).$$

GNN in time-invariant case

Additional dynamic state equations for outer inverses

- One RNN model is based on the representation

$$A^{(2)}_{\mathcal{R}(G),\mathcal{N}(G)} = G(AG)^{\#} = (GA)^{\#}G$$

of the outer inverse $A^{(2)}_{\mathcal{R}(G),\mathcal{N}(G)}$ and uses the GNNAD-I model in order to compute $(AG)^{\#}$ or $(GA)^{\#}$.
State equation:

$$\frac{dV(t)}{dt} = -\beta \left((GA)^{l+1}V(t) - (GA)^{l}\right), l \geq \mathrm{ind}(A).$$

Output equation:

$$X(t) = V(t)\, G.$$

GNN in time-invariant case

Additional dynamic state equations for outer inverses

- But, this approach requires an appropriate power of AG or GA.

GNN in time-invariant case

Additional dynamic state equations for outer inverses

- But, this approach requires an appropriate power of AG or GA.
- Moreover, this approach is not globally asymptotically stable, and requires the zero initial approximation $V_0 = 0$.

GNN in time-invariant case

Additional dynamic state equations for outer inverses

- An additional dynamic equation and the induced RNN were proposed in
  [P.S. Stanimirović, I. Živković, Y. Wei, *Neural network approach to computing outer inverses based on the full rank representation*, Linear Algebra Appl., **501** (2016), 344–362.]
  They were derived using the representation

$$A^{(2)}_{\mathcal{R}(G),\mathcal{N}(G)} = A^{(2)}_{\mathcal{R}(P),\mathcal{N}(Q)} = P(QAP)^{-1}Q, \quad G = PQ,$$

  where

$$P \in \mathbb{R}^{n\times s}_{s}, \ Q \in \mathbb{R}^{s\times m}_{s}.$$

GNN in time-invariant case

Additional dynamic state equations for outer inverses

- State equation:

$$\frac{dV(t)}{dt} = -\beta \left((QAP)^{\mathrm{T}} QAP\, V(t) - (QAP)^{\mathrm{T}} \right), \; V(0) = V_0,$$

GNN in time-invariant case

Additional dynamic state equations for outer inverses

- State equation:

$$\frac{dV(t)}{dt} = -\beta \left((QAP)^{\mathrm{T}} QAP\, V(t) - (QAP)^{\mathrm{T}} \right), \; V(0) = V_0,$$

Output equation:

$$X(t) = P\, V(t)\, Q = A^{(2)}_{\mathcal{R}(P),\mathcal{N}(Q)}.$$

GNN in time-invariant case

Additional dynamic state equations for outer inverses

- This approach resolves the problems of restrictions on the spectrum as well as the problem of cumulative matrix multiplications.

GNN in time-invariant case

Additional dynamic state equations for outer inverses

- This approach resolves the problems of restrictions on the spectrum as well as the problem of cumulative matrix multiplications.

  Firstly, the proposed neural network possesses the global stability, since QAP is invertible.

GNN in time-invariant case

Additional dynamic state equations for outer inverses

- This approach resolves the problems of restrictions on the spectrum as well as the problem of cumulative matrix multiplications.

  Firstly, the proposed neural network possesses the global stability, since QAP is invertible.

  Moreover, this approach is fastest, since it uses RNN for matrices of small dimensions $s \times s$, $s \leq \min\{m, n\}$.

GNN in time-invariant case

Symbolic computation of outer inverses using dynamic state equation

- An algorithm for symbolic computation of outer inverses is defined by means of the exact solution of first order systems of differential equations which appear in the dynamic state equation GNNATS2-I.

GNN in time-invariant case

Symbolic computation of outer inverses using dynamic state equation

- An algorithm for symbolic computation of outer inverses is defined by means of the exact solution of first order systems of differential equations which appear in the dynamic state equation GNNATS2-I.
- The algorithm is applicable to matrices whose elements are integers, rational numbers as well as rational or polynomial expressions.

GNN in time-invariant case

Symbolic computation of outer inverses using dynamic state equation

- Main algorithmic steps:

GNN in time-invariant case

Symbolic computation of outer inverses using dynamic state equation

- Main algorithmic steps:
  1. Solve dynamical equation elementwise and symbolically. Denote the output by V_G.

GNN in time-invariant case

Symbolic computation of outer inverses using dynamic state equation

- Main algorithmic steps:
  1. Solve dynamical equation elementwise and symbolically. Denote the output by V_G.
  2. Compute elementwise

$$\overline{V_G} = \lim_{t \to \infty} V_G(t) \approx A^{(2)}_{\mathcal{R}(G),\mathcal{N}(G)}.$$

GNN in time-invariant case

Symbolic computation of outer inverses using dynamic state equation

- Main algorithmic steps:
  1. Solve dynamical equation elementwise and symbolically. Denote the output by V_G.
  2. Compute elementwise

$$\overline{V_G} = \lim_{t \to \infty} V_G(t) \approx A^{(2)}_{\mathcal{R}(G),\mathcal{N}(G)}.$$

- The implementation is performed by the following code in the symbolic mathematical computation program *Mathematica*.

GNN in time-invariant case

Symbolic computation of outer inverses using dynamic state equation

- Main algorithmic steps:
  1. Solve dynamical equation elementwise and symbolically. Denote the output by V_G.
  2. Compute elementwise

$$\overline{V_G} = \lim_{t \to \infty} V_G(t) \approx A^{(2)}_{\mathcal{R}(G),\mathcal{N}(G)}.$$

- The implementation is performed by the following code in the symbolic mathematical computation program *Mathematica*.
  - Main *Mathematica* functions used are *DSolve* and *Limit*.

ZNN in time-varying case

Complex ZNN models for time-varying Drazin inverse

Lemma 1

Let $A \in \mathbb{C}^{n\times n}$ be a given square matrix. A closed-form solution of A^{D} is given by

$$A^{\mathrm{D}} = \lim_{\lambda\to 0} \left(A^l (A^{2l+1})^{\mathrm{H}} A^{l+1} + \lambda I\right)^{-1} A^l (A^{2l+1})^{\mathrm{H}} A^l, \quad l \geq \mathrm{Ind}(A).$$

ZNN in time-varying case

Complex ZNN models for time-varying Drazin inverse

Lemma 1

Let $A \in \mathbb{C}^{n\times n}$ be a given square matrix. A closed-form solution of A^{D} is given by

$$A^{\mathrm{D}} = \lim_{\lambda\to 0} \left(A^l (A^{2l+1})^{\mathrm{H}} A^{l+1} + \lambda I\right)^{-1} A^l (A^{2l+1})^{\mathrm{H}} A^l, \quad l \geq \mathrm{Ind}(A).$$

- **Step 1.** (Choose a suitable ZF).

$$E(t) = (G(t)A(t) + \lambda I)\, V(t) - G(t),$$

where

$$G(t) = A(t)^l \left(A(t)^{2l+1}\right)^{\mathrm{H}} A(t)^l.$$

ZNN in time-varying case

Complex ZNN models for time-varying Drazin inverse, model LZNN-I

- **Step 2.** (Define Zhang design formula).

$$\dot{E}(t) := \frac{\mathrm{d}E(t)}{\mathrm{d}t} = -\gamma \mathcal{H}(E(t)),$$

where the design parameter $\gamma \in \mathbb{R}$, $\gamma > 0$, corresponds to the inductance parameter, and $\mathcal{H}(\cdot)$ is a complex-valued activation function.

ZNN in time-varying case

Complex ZNN models for time-varying Drazin inverse, model LZNN-I

- **Step 3.** (Generate a ZNN model LZNN-I).

$$\dot{V}(t) = \frac{1}{\lambda}\Big[-G(t)A(t)\dot{V}(t) + \dot{G}(t) - \left(\dot{G}(t)A(t) + G(t)\dot{A}(t)\right)V(t) - \gamma\mathcal{H}\left(\left(G(t)A(t) + \lambda I\right)V(t) - G(t)\right)\Big].$$

ZNN in time-varying case

Complex ZNN models for time-varying Drazin inverse, model LZNN-I

- **Step 3.** (Generate a ZNN model LZNN-I).

$$\dot{V}(t) = \frac{1}{\lambda}\Big[-G(t)A(t)\dot{V}(t) + \dot{G}(t) - \left(\dot{G}(t)A(t) + G(t)\dot{A}(t)\right)V(t) - \gamma\mathcal{H}\left(\left(G(t)A(t) + \lambda I\right)V(t) - G(t)\right)\Big].$$

- The LZNN-I model was proposed in [X.-Z. Wang, Y. Wei, P.S. Stanimirović, *Complex neural network models for time-varying Drazin inverse*, Neural Computation, Accepted for publication.]

ZNN in time-varying case

Complex Neural Network Models for Time-Varying Drazin Inverse, model LZNN-I

- The dynamic equation of the (ij)th neuron can be presented in the form

$$\dot{v}_{ij} = \frac{1}{\lambda}\left(\sum_{k=1}^{n} b_{ik}\dot{v}_{kj} - \gamma\sum_{k=1}^{n}\mathcal{H}\left(c_{ik}v_{kj} - g_{ij}\right) - \sum_{k=1}^{n} d_{ik}v_{kj} + \dot{g}_{ij}\right),$$

where b_{ik}, c_{ik}, d_{ik} and g_{ij} denote the elements of the matrices that appear in the LZNN-I model:
$B = -G(t)A(t)$, $C = G(t)A(t) + \lambda I$,
$D = \dot{G}(t)A(t) + G(t)\dot{A}(t)$ and $G(t)$, respectively.

ZNN in time-varying case

Complex Neural Network Models for Time-Varying Drazin Inverse, model LZNN-I

- The dynamic equation of the (ij)th neuron can be presented in the form

$$\dot{v}_{ij} = \frac{1}{\lambda}\left(\sum_{k=1}^{n} b_{ik}\dot{v}_{kj} - \gamma\sum_{k=1}^{n}\mathcal{H}\left(c_{ik}v_{kj} - g_{ij}\right) - \sum_{k=1}^{n} d_{ik}v_{kj} + \dot{g}_{ij}\right),$$

where b_{ik}, c_{ik}, d_{ik} and g_{ij} denote the elements of the matrices that appear in the LZNN-I model:
$B = -G(t)A(t)$, $C = G(t)A(t) + \lambda I$,
$D = \dot{G}(t)A(t) + G(t)\dot{A}(t)$ and $G(t)$, respectively.

- Note that the parameter λ should be sufficiently close to zero.

ZNN in time-varying case

Complex ZNN models for time-varying Drazin inverse, model LZNN-I

Definition 2

Let $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times n}$ be given.

ZNN in time-varying case

Complex ZNN models for time-varying Drazin inverse, model LZNN-I

Definition 2

Let $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times n}$ be given.Type I activation function array is defined by

$$\mathcal{H}_1(A+\iota B) = \mathcal{F}(A) + \iota\mathcal{F}(B),\ \iota = \sqrt{-1}.$$

ZNN in time-varying case

Complex ZNN models for time-varying Drazin inverse, model LZNN-I

Definition 2

Let $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times n}$ be given.Type I activation function array is defined by

$$\mathcal{H}_1(A+\iota B) = \mathcal{F}(A) + \iota\mathcal{F}(B),\ \iota = \sqrt{-1}.$$

Type II activation function array is defined as

$$\mathcal{H}_2(A+\iota B) = \mathcal{F}(\Gamma)\circ \exp(\iota\Theta),$$

where

$\Gamma = |A+\iota B| \in \mathbb{R}^{n\times n}$ and $\Theta = \Theta(A+\iota B) \in (-\pi,\pi]^{n\times n}$ denote element-wise modulus and the element-wise arguments, respectively, of the complex matrix $A+\iota B$.

ZNN in time-varying case

Complex ZNN models for time-varying Drazin inverse, model LZNN-I

Additionally, $\mathcal{F}(A)$ is defined to be an element-wise odd and monotonically increasing function for a real matrix A, i.e., $\mathcal{F}(A) = (f(a_{kj}))$ for $A = (a_{kj}) \in \mathbb{R}^{n \times n}$, with an odd and monotonically increasing function $f(\cdot)$.

ZNN in time-varying case

Complex ZNN models for time-varying Drazin inverse, model LZNN-II

Lemma 3

The Drazin inverse of $A \in \mathbb{C}^{n\times n}$ possesses the limit representation

$$A^{\mathrm{D}} = \lim_{\lambda\to 0} (A+\lambda I)^{-(l+1)} A^l, \ \ l \geq \mathrm{Ind}(A).$$

ZNN in time-varying case

Complex ZNN models for time-varying Drazin inverse, model LZNN-II

Lemma 3

The Drazin inverse of $A \in \mathbb{C}^{n\times n}$ possesses the limit representation

$$A^{\mathrm{D}} = \lim_{\lambda\to 0} (A+\lambda I)^{-(l+1)} A^l, \ \ l \geq \mathrm{Ind}(A).$$

- The matrix $(A(t)+\lambda I)^{l+1}$, $\lambda > 0$, is nonsingular for any complex (regular or singular) matrix $A(t)$.

ZNN in time-varying case

Complex ZNN models for time-varying Drazin inverse, model LZNN-II

Lemma 3

The Drazin inverse of $A \in \mathbb{C}^{n\times n}$ possesses the limit representation

$$A^{\mathrm{D}} = \lim_{\lambda\to 0} (A+\lambda I)^{-(l+1)} A^l, \ \ l \geq \mathrm{Ind}(A).$$

- The matrix $(A(t)+\lambda I)^{l+1}$, $\lambda > 0$, is nonsingular for any complex (regular or singular) matrix $A(t)$.
  Therefore,

$$\lim_{\lambda\to 0} (A+\lambda I)^{l+1} A^{\mathrm{D}} = A^l, \ \ l \geq \mathrm{Ind}(A).$$

ZNN in time-varying case

Complex ZNN models for time-varying Drazin inverse, model LZNN-II

- It is possible to define the following ZF function as the fundamental error-monitoring function (called LZF2):

$$E_1(t) = (A(t) + \lambda I)^{l+1} V(t) - A(t)^l \triangleq H(t)V(t) - Q(t),$$

where $H(t) = (A(t) + \lambda I)^{l+1}$ and $Q(t) = A(t)^l$.

ZNN in time-varying case

Complex ZNN models for time-varying Drazin inverse, model LZNN-II

- It is possible to define the following ZF function as the fundamental error-monitoring function (called LZF2):

$$E_1(t) = (A(t) + \lambda I)^{l+1} V(t) - A(t)^l \triangleq H(t)V(t) - Q(t),$$

where $H(t) = (A(t) + \lambda I)^{l+1}$ and $Q(t) = A(t)^l$.

- The time derivative of $E_1(t)$ is equal to

$$\dot{E}_1(t) = -\dot{Q}(t) + H(t)\dot{V}(t) + \dot{H}(t)V(t).$$

ZNN in time-varying case

Complex ZNN models for time-varying Drazin inverse, model LZNN-II

- Following the design formula

$$\dot{E}(t) := \frac{\mathrm{d}E(t)}{\mathrm{d}t} = -\gamma\mathcal{H}(E(t)),$$

we obtain the implicit dynamic equation which initiates the second complex neural network model LZNN-II as

$$H(t)\dot{V}(t) = \dot{Q}(t) - \dot{H}(t)V(t) - \gamma\mathcal{H}\left((H(t)V(t) - Q(t))\right).$$

Time-varying case

Complex ZNN models for time-varying Drazin inverse, model LZNN-II

Theorem 2

Given a time-varying complex matrix $A(t) \in \mathbb{C}^{n\times n}$. *If an activation function with an odd and monotonically increasing function* $f(\cdot)$ *is used, then the state matrix* $V(t) \in \mathbb{C}^{n\times n}$ *of the complex neural network model* LZNN-II *globally converges to* $A(t)^{\mathrm{D}}$, *starting from arbitrary initial state* $V(0)$.

Time-varying case

Complex ZNN models for time-varying Drazin inverse, model LZNN-II

Theorem 2

Given a time-varying complex matrix $A(t) \in \mathbb{C}^{n\times n}$. *If an activation function with an odd and monotonically increasing function* $f(\cdot)$ *is used, then the state matrix* $V(t) \in \mathbb{C}^{n\times n}$ *of the complex neural network model* LZNN-II *globally converges to* $A(t)^{\mathrm{D}}$, *starting from arbitrary initial state* $V(0)$.

The LZNN-II and LZNN-II models were proposed in [X.-Z. Wang, Y. Wei, P.S. Stanimirović, *Complex neural network models for time-varying Drazin inverse*, Neural Computation, Accepted for publication.]

Time-varying case

RNN approach based on constrained quadratic optimization

- Two continuous-time neural networks for computing generalized inverses of complex-valued matrices were presented in
  [Y. Xia, P.S. Stanimirović, S. Zhang, *Neural network for computing pseudoinverses and outer inverses of complex-valued matrices*, Appl. Math. Comput. **273** (2016), 1107–1121.]

Time-varying case

RNN approach based on constrained quadratic optimization

- Two continuous-time neural networks for computing generalized inverses of complex-valued matrices were presented in
  [Y. Xia, P.S. Stanimirović, S. Zhang, *Neural network for computing pseudoinverses and outer inverses of complex-valued matrices*, Appl. Math. Comput. **273** (2016), 1107–1121.]
- These neural networks are generated using the fact that the outer inverse $A^{(2)}_{T,S}$ and the Moore-Penrose inverse can be derived as the solution of appropriate, matrix valued, convex quadratic programming problems.

Time-varying case

RNN approach based on constrained quadratic optimization

- $A^{\dagger}$ is a solution of the following optimization problem with respect to $X \in \mathbb{C}^{n \times m}$

$$\begin{aligned} &\text{minimize} \quad \|X\|_F^2 \\ &\text{subject to} \quad A^H A X = A^H, \end{aligned}$$

Time-varying case

RNN approach based on constrained quadratic optimization

- $A^{\dagger}$ is a solution of the following optimization problem with respect to $X \in \mathbb{C}^{n \times m}$

$$\begin{aligned} &\text{minimize} \quad \|X\|_F^2 \\ &\text{subject to} \quad A^H A X = A^H, \end{aligned}$$

- The proposed continuous-time neural networks have a low complexity of implementation and they are globally convergent without any condition.

Time-varying case

RNN approach based on constrained quadratic optimization

Continuous-time algorithm (I) for $m \geq n$ case

State equation

$$\frac{dY(t)}{dt} = \beta \left[A^{\mathrm{H}} - (A^{\mathrm{H}} A)^2 Y(t) \right].$$

Output equation

$$X(t) = A^H A Y(t).$$

$Y(t) \in \mathbb{C}^{n \times m}$ is the state matrix trajectory, $X(t) \in \mathbb{C}^{n \times m}$ is the output matrix trajectory, and $\beta > 0$ is a scaling constant.