Contents

1 Abstract

2 Introduction

When studying systems and control theory we often focus on finding properties of a given mathematical model. However, in the real world the mathematical model of a system is not always available. In most cases we only have the measurements/data as produced by a system. To combat this, we can use tools from the field of data-driven analysis and control. These tools are able to infer system theoretical properties from the data without explicitly knowing the underlying mathematical model of the system. In this paper we will look at the implementation of the tools provided in [**waarde2019data**] and [**waarde2020noisy**].

3 Definitions

In this section we will introduce the notion of informativity. We will also introduce ourselves with the notation that will be used in the paper to indicate data and systems.

3.1 Informativity

Let Σ be the set of discrete time models with a given state space dimension n and a given input space dimension m. Then if we have state/input data $\mathcal{D}$ of this form, then we know that the system $\mathcal{S}$ that produced this data is contained in the set Σ. However, using Σ is not very useful unless we reduce the set to be more manageable, we can do this by using our knowledge of the true system, namely the data. We will only consider systems that are able to produce the data, we will call this set $\Sigma_{\mathcal{D}} \subseteq \Sigma$.

Suppose we want to show if the true system is controllable, we might not be able to uniquely identify the true system using the data, i.e. $\#\Sigma_{\mathcal{D}} > 1$. However if we are able to show that every system in the set $\Sigma_{\mathcal{D}}$ is controllable, then we would also know that the true system is controllable. This is the idea behind data informativity. We say the data $\mathcal{D}$ is informative for a property $\mathcal{P}$ if all systems that describe the data $\Sigma_{\mathcal{D}}$ have the property $\mathcal{P}$. Let $\Sigma_{\mathcal{P}} \subseteq \Sigma$ be the set of all systems that have the property $\mathcal{P}$. Then we can reformulate the definition of data informativity as follows.

Def: Informativity [waarde2019data]
We say that the data $\mathcal{D}$ is informative for a property $\mathcal{P}$ if $\Sigma_{\mathcal{D}} \subseteq \Sigma_{\mathcal{P}}$.

Suppose that the data describes only the true system, i.e. $\Sigma_{(U_-,X)} = \{\mathcal{S}\}$ then we know that if a property $\mathcal{P}$ hold for $\mathcal{S}$ then the data is informative for that property. However, in general, just because $\mathcal{S}$ has a property $\mathcal{P}$ does not immediately imply that the data is informative for $\mathcal{P}$ since the data might describe more then one system. Later on we will see this in an example where the data describes infinity many systems.

add reference to example with no identification and state feedback

We will also focus on control problems. Suppose we want to see if the property 'is stable in full state feedback with a controller K' holds on the data, then we need to know if all systems are stabilisable by state feedback using the controller K. For this we will define the set of systems that are stabilised using state feedback for the controller K as follows:

$$\Sigma_K = \{(A, B) \,|\, A + B\,K \text{ is stable}\}$$

Then we have that the data is informative if $\Sigma_{\mathcal{D}} \subseteq \Sigma_K$. We will generalise this using the following definition.

Def: Informativity for control [waarde2019data]
We say that the data $\mathcal{D}$ is informative for a property $\mathcal{P}(\cdot)$ if there exists a controller $\mathcal{K}$ such that $\Sigma_{\mathcal{D}} \subseteq \Sigma_{\mathcal{P}(\mathcal{K})}$.

3.2 Data and model classes

We will use the following example [**waarde2019data**] to give a more precise definition of data.

Let n be the dimension of the state space and m be the dimension of the input space. Assume that both are known. Then we know that all systems contained in Σ are of the form:

$$\mathbf{x}(t+1) = A\mathbf{x}(t) + B\mathbf{u}(t) \tag{1}$$

Where $\mathbf{x}(t)$ is the n-dimensional state vector and $\mathbf{u}(t)$ is the m-dimensional input vector evaluated at time t. We will assume our data is generated by the 'true' system, we will denote this system as (A_s, B_s). We will now measure the input and state data on q time intervals $\{0, 1, \ldots, T_i\}$ where $i \in \{1, 2, \ldots, q\}$. We denote the data collected on one of these intervals as follows:

$$U_-^i = \begin{bmatrix} u^i(0) & u^i(1) & \ldots & u^i(T_i - 1) \end{bmatrix}$$
$$X^i = \begin{bmatrix} x^i(0) & x^i(1) & \ldots & x^i(T_i) \end{bmatrix}$$

We will now 'split' the state data into a 'past' and 'future' segment, these are defined similar to U_-^i.

$$X_-^i = \begin{bmatrix} x^i(0) & \ldots & x^i(T_i - 1) \end{bmatrix}$$
$$X_+^i = \begin{bmatrix} x^i(1) & \ldots & x^i(T_i) \end{bmatrix}$$

With this representation of our state and input data we have that $X_+^i = A_s\,X_-^i + B_s\,U_-^i$. This holds for all measured intervals i of the true system. We will now combine the data of all intervals to get a more general form.

$$U_- = \begin{bmatrix} U_-^1 & \ldots & U_-^q \end{bmatrix} \qquad X = \begin{bmatrix} X^1 & \ldots & X^q \end{bmatrix}$$
$$X_+ = \begin{bmatrix} X_+^1 & \ldots & X_+^q \end{bmatrix} \qquad X_- = \begin{bmatrix} X_-^1 & \ldots & X_-^q \end{bmatrix}$$

We will define our data $\mathcal{D} := (U_-, X)$. In this example we have that $\Sigma_{\mathcal{D}} = \Sigma_{(U_-,X)} = \Sigma_{i/s}$ and is defined by:

$$\Sigma_{(U_-,X)} = \left\{ (A, B) \,|\, X_+ = \begin{bmatrix} A & B \end{bmatrix} \begin{bmatrix} X_- \\ U_- \end{bmatrix} \right\}$$

By construction we know that at least the true system (A_s, B_s) is contained in this set.

We can extend this concept to also include the output of a system. Assume we have a system of the form:

$$\mathbf{x}(t+1) = A\mathbf{x}(t) + B\mathbf{u}(t) \tag{2}$$
$$\mathbf{y}(t+1) = C\mathbf{x}(t) + D\mathbf{u}(t) \tag{3}$$

Let us define Y_- in the following way:

$$Y_-^i = \begin{bmatrix} y^i(0) & y^i(1) & \dots & y^i(T_i - 1); \end{bmatrix}$$
$$Y_- = \begin{bmatrix} Y_-^1 & Y_-^2 & \dots & Y_-^q \end{bmatrix}$$

Then we can define the set of systems that can describe the data as follows:

$$\Sigma_{(U_-,X,Y_-)} = \left\{ (A, B, C, D) \,|\, \begin{bmatrix} X_+ \\ Y_- \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} X_- \\ U_- \end{bmatrix} \right\} \tag{4}$$

4 System identification

In this section we will see how we can use the data to identify the true system. We will first consider the mathematics and then we will see how this can be implemented. Lastly we will consider an example that we will solve using the Matlab function based on the previously discussed algorithm.

Def: Informative for system identification [waarde2019data]
We say that the data is informative for system identification if the data only describes 1 system, i.e. $\Sigma_{\mathcal{D}} = \{(A_s, B_s)\}$.

4.1 Mathematics

Consider systems of the form (1) and let (U_-, X) be the data generated by the true system. Assume that we have recorded T data points, i.e. the dimension of U_- is $m \times T$ and the dimension of X is $n \times (T+1)$. Recall that the set of systems described by this data is given by:

$$\Sigma_{(U_-,X)} = \left\{ (A,B) \,|\, X_+ = \begin{bmatrix} A & B \end{bmatrix} \begin{bmatrix} X_- \\ U_- \end{bmatrix} \right\} \tag{5}$$

Where $\begin{bmatrix} X_- \\ U_- \end{bmatrix}$ is $(n+m) \times T$. Suppose that the *rank* $\left(\begin{bmatrix} X_- \\ U_- \end{bmatrix} \right) = n + m$, then there exists a right inverse $\begin{bmatrix} V_1 & V_2 \end{bmatrix}$ such that $\begin{bmatrix} X_- \\ U_- \end{bmatrix} * \begin{bmatrix} V_1 & V_2 \end{bmatrix} = I_{n+m}$. If we multiply the equation from (5) with this inverse we get the following:

$$X_+ \begin{bmatrix} V_1 & V_2 \end{bmatrix} = \begin{bmatrix} A & B \end{bmatrix} I_{n+m}$$

Thus if the data is full rank we are able to retrieve the (A, B) pair directly. This results in the following proposition.

4.2 Algorithm

Prop: [waarde2019data]
The data (U_-, X) *is informative for system identification if and only if*

$$rank \left(\begin{bmatrix} X_- \\ U_- \end{bmatrix} \right) = n + m$$

Furthermore, if the data is full rank, there exists an right inverse $\begin{bmatrix} V_1 & V_2 \end{bmatrix}$ *(as defined above), and for any such right inverse* $A_s = X_+ V_1$ *and* $B_s = X_+ V_2$.

Suppose we are considering a system without any inputs, then the proposition reduces to checking if X_- has full row rank and finding a right inverse $X_-^{\dagger}$ of X_-. Then we retrieve the system as follows $A_s = X_+ X_-^{\dagger}$.

Lets assume we are considering an input, state, output system of the form (2):

Recall the set of systems described by the data is given by (4):

$$\Sigma_{(U_-,X,Y_-)} = \left\{ (A,B,C,D) \,|\, \begin{bmatrix} X_+ \\ Y_- \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} X_- \\ U_- \end{bmatrix} \right\}$$

In this case if the proposition holds, we can also retrieve the C and D matrix by computing $C = Y\_V_1$ and $D = Y\_V_2$.

4.3 Examples using implementation

The algorithm above is implemented in the following functions:

Syntax

```
[bool, A] = isInformIdentification(X)
[bool, A, B] = isInformIdentification(X, U)
[bool, A, B, C, D] = isInformIdentification(X, U, Y)
```

Description

`[bool, A] = isInformIdentification(X)`: returns if the state data is informative for system identification, if it is then A contains the A is the system matrix in state space representation.
`[bool, A, B] = isInformIdentification(X, U)`: returns if the state and input data is informative for system identification, if it is then A and B are the system matrices in state space representation.
`[bool, A, B, C, D] = isInformIdentification(X, U, Y)`: returns if the state, input and output data is informative for system identification, if it is then A, B, C and D are the system matrices in state space representation.

Input arguments

`X`: State data matrix of dimension $n \times T + 1$.
`U`: Input data matrix of dimension $m \times T$.
`Y`: Output data matrix of dimension $p \times T$.

Output arguments

`bool`: (boolean) True if the data is informative for system identification, false otherwise
`A`: (matrix) If the data is informative, it contains the systems A matrix, empty otherwise.
`B`: (matrix) If the data is informative, it contains the systems B matrix, empty otherwise.
`C`: (matrix) If the data is informative, it contains the systems C matrix, empty otherwise.
`D`: (matrix) If the data is informative, it contains the systems D matrix, empty otherwise.

4.3.1 Examples

Lets consider the following state and input data:

$$X = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad U = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

As we can see the data is not sufficient for system identification:

$$rank\left(\begin{bmatrix}X_-\\U_-\end{bmatrix}\right) = rank\left(\begin{bmatrix}0 & 1\\0 & 0\\1 & 0\end{bmatrix}\right) = 2 \neq 3$$

This is because the data can be generated by systems of the following form:

$$\Sigma_{i/s} = \left\{\left(\begin{bmatrix}0 & a_1\\1 & a_2\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}\right) \mid a_1, a_2 \in \mathbb{R}\right\}$$

However, if we where to consider the same data with 1 additional data point then the data would be informative for system identification:

$$X_- = \begin{bmatrix}0 & 1 & 0\\0 & 0 & 1\end{bmatrix} \qquad U = \begin{bmatrix}1 & 0 & \alpha\end{bmatrix} \qquad rank\left(\begin{bmatrix}0 & 1 & 0\\0 & 0 & 1\\1 & 0 & \alpha\end{bmatrix}\right) = 3$$

We can also find the same results using the MatLab functions:

```
X = [0 1 0 ; 0 0 1]; U = [1 0];
[bool, A, B] = isInformIdentification(X, U)
```

Which will return: `[ false, [], [] ]`.

5 Controllability

In this section we will see how we can use the data to conclude if a set of systems are controllable of not. We will first consider the mathematics and then we will see how this can be implemented. Lastly we will consider an example that we will solve using the Matlab function based on the previously discussed algorithm.

Def: Informative for controllability [waarde2019data]
We say that the data is informative for controllability if all systems that describe the data are controllable. I.e. $\Sigma_{i/o} \subseteq \{(A, B) \,|\, (A, B) \text{ is controllable}\}$.

We will base our algorithm on theorem 8 and remark 9 from [**waarde2019data**]. These give necessary and sufficient conditions for the informativity.

Thr: Informative for controllability
The data (U_-, X) *is informative for controllability if and only if* $rank(X_+ - \lambda X_-) = n \; \forall \lambda \in \mathbb{C}$.

This theorem can be reduced to check a finite amount of complex numbers similar to the Hautus test. We have that the above theorem is equivalent to $rank(X+) = n$ and $rank(X_+ - \lambda X_-) = n$ for all $\lambda \neq 0$ with $\lambda^{-1} \in \sigma(X_- X_+^\dagger)$ with $X_+^\dagger$ being the right inverse of X_+ and $\sigma(\cdot)$ denotes the set of eigenvalues of the matrix. proof?

5.1 Examples using implementation

The algorithm above is implemented in the following functions:

Syntax

`[bool] = isInformControllable(X)`

Description

`[bool] = isInformControllable(X)`: Returns if the state data from an input-state dataset is informative for controllability.

Input arguments

`X`: State data matrix of dimension $n \times T + 1$ from an input-state dataset.

Output arguments

`bool`: (boolean) True if the data is informative for controllability, false otherwise

Examples

We will consider the same data as we did in the example of system identification. Recall the provided data was:

$$X = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad U = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

For the data to be informative for controllability we need that $X_+ - \lambda X_-$ is full row rank for all $\lambda \in \mathbb{C}$.

$$rank(X_+ - \lambda X_-) = rank\left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 & \lambda \\ 0 & 0 \end{bmatrix} \right) = 2$$

Hence the data is informative for controllability.

We can also find the same result by using the Matlab function:

```
X = [0 1 0 ; 0 0 1]; U = [1 0];
[bool] = isInformControllable(X)
```

Which will return: `[ 1 ]`.

We can verify the result by considering the systems that generate this data. Recall that the data is generated by systems of the form:

$$\Sigma_{i/s} = \left\{ \left(\begin{bmatrix} 0 & a_1 \\ 1 & a_2 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right) \mid a_1, a_2 \in \mathbb{R} \right\}$$

Thus the controllability matrix is given by:

$$\begin{bmatrix} B & AB \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Since the controllability matrix has full rank we can conclude that systems of this form are indeed controllable.

6 Stabilisability

In this section we will see how we can use the data to conclude if a set of systems are stabilisable of not. We will first consider the mathematics and then we will see how this can be implemented. Lastly we will consider an example that we will solve using the Matlab function based on the previously discussed algorithm.

Def: Informative for stabilisability [waarde2019data]
We say that the data is informative for stabilisability if all systems that describe the data are stabilisable. I.e. $\Sigma_{i/o} \subseteq \{(A,B) \mid (A,B) \text{ is stabilisable}\}$.

We will base our algorithm on theorem 8 and remark 9 from [**waarde2019data**]. These give necessary and sufficient conditions for the informativity.

Thr: Informative for stabilisability
The data (U_-, X) *is informative for stabilisability if and only if* $rank(X_+ - \lambda X_-) = n \ \forall \lambda \in \mathbb{C}$ *such that* $|\lambda| \geq 1$.

This theorem can be reduced to check a finite amount of complex numbers similar to the Hautus test. We have that the above theorem is equivalent to $rank(X_+ - X_-) = n$ and $rank(X_+ - \lambda X_-) = n$ for all $\lambda \neq 1$ with $(\lambda - 1)^{-1} \in \sigma(X_-(X_+ - X_-)^\dagger)$ with $(X_+ - X_-)^\dagger$ being the right inverse of $(X_+ - X_-)$ and $\sigma(\cdot)$ denotes the set of eigenvalues of the matrix.

proof?

6.1 Examples using implementation

The algorithm above is implemented in the following functions:

Syntax

```
[bool] = isInformStabilisable(X)
[bool] = isInformStabilisable(X, tolerance)
```

Description

`[bool] = isInformStabilisable(X)`: Returns if the data is informative for stabilisability. Uses default tolerance of `1e-14`.
`[bool] = isInformStabilisable(X, tolerance)`: Returns if the data is informative for stabilisability given a specific tolerance.

Input arguments

`X`: State data matrix of dimension $n \times T + 1$.
`tolerance`: Tolerance used for determining when a value is zero up to machine precision. Default value is `1e-14`.

Output arguments

`bool`: (boolean) True if the data is informative for stabilisability, false otherwise

6.1.1 Examples

In this example we will consider the same data that we used in system identification and controllability, namely:

$$X = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad\qquad U = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

For the data to be informative for stabilisability we need that $X_+ - \lambda X_-$ is full row rank for all $\lambda \in \mathbb{C}$ such that $|\lambda| \geq 1$.

$$rank(X_+ - \lambda X_-) = rank\left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 & \lambda \\ 0 & 0 \end{bmatrix}\right) = 2$$

Since the rank condition hold for all λ it will also hold for all $|\lambda| \geq 1$. Hence the data is informative for stabilisability. This is consistent with the non data driven theory that controllability implies stabilisability.

We can also find the same result by using the Matlab function:

```
X = [0 1 0 ; 0 0 1]; U = [1 0];
[bool] = isInformStabilisable(X)
```

Which will return: `[ 1 ]`.

We can verify the result by considering the systems that generate this data. Recall that the data is generated by systems of the form:

$$\Sigma_{i/s} = \left\{ \left(\begin{bmatrix} 0 & a_1 \\ 1 & a_2 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right) \mid a_1, a_2 \in \mathbb{R} \right\}$$

Recall that a pair (A, B) is stabilisable if and only if $\begin{bmatrix} A - \lambda I & B \end{bmatrix}$ is full row rank for all $\lambda \in \sigma(A)$ such that $Re(\lambda) \geq 0$.

$$rank\left(\begin{bmatrix} A - \lambda I & B \end{bmatrix}\right) = rank\left(\begin{bmatrix} -\lambda & a_1 & 1 \\ 1 & a_2 - \lambda & 0 \end{bmatrix}\right) = 2$$

Thus the systems that describe the data are stabilisable.

7 Stability

In this section we will consider data informativity for stability of unforced systems. We will first see how we can define this notion after which we will show how we can reduce it to a program friendly condition.

In this section we will consider our data to be obtained from an unforced system i.e. there was no input. Because of this we will define our data and set of systems as follows:

$$\mathcal{D} = (X) \qquad\qquad \Sigma_{\mathcal{D}} = \{A | X_+ = A X_-\}$$

Using these definitions we can define data informativity for stability as follows.

Def: Informative for stability
We say the data is informative for stability if all unforced systems describing the data are stable.

7.1 Mathematics stability

However, using our definition of informativity for stabilisability we are able to reduce the condition to showing that the data is informative for identification and that the identified system is stable.

Cor: [Cor 11]waarde2019data
The data X is informative for stability if and only if X_- has full row rank and $X_+X_-^\dagger$ is stable for any right inverse $X_-^\dagger$. This is equivalent to $\Sigma_{\mathcal{D}} = \{A_s\}$ and $A_s = X_+X_-^\dagger$ being stable.

Proof '$\Rightarrow$':

Proof '$\Leftarrow$':

7.2 Examples using implementation

The algorithm above is implemented in the following functions:

Syntax

`[bool] = isInformStable(X)`

Description

`[bool] = isInformStable(X)`: Returns if the data of a unforced system is informative for stability.

Input arguments

`X`: State data matrix of dimension $n \times T + 1$.

Output arguments

`bool`: (boolean) True if the data is informative for stability, false otherwise

7.2.1 Examples

For this example we will consider the following state data generated by a unforced system:

$$X = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{4} \\ 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

For the data to be informative for stability we need that the data is informative for system identification and that the identified system is stable. The data is informative for system identification if and only if there exists an right inverse $X_-^\dagger$ of X_-. Then the identified system is given by $A = X_+ X_-^\dagger$.

$$A = \begin{bmatrix} \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} 1 & \frac{1}{2} \\ 0 & \frac{1}{2} \end{bmatrix}^{-1} = \begin{bmatrix} \frac{1}{2} & 0 \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

As we can see from the A matrix its eigenvalues are $\sigma(A) = \{\frac{1}{2}, \frac{1}{2}\}$. Thus the system is stable and hence the data is informative for stability.

We can also find the same result by using the Matlab function:

```
X = [1 0.5 0.25; 0 0.5 0.5];
[bool, A] = isInformIdentification(X)
```

Which will return: `[ 1, [0.5 0 ; 0.5 0.5] ]`.

8 State feedback

In this section we will see when en how we can construct a state feedback controller directly from the state-input data. We will first consider the mathematics after which we will look at how we can implement this. Lastly we will look at an example and how we can apply the provided functions.

In this section we will consider closed loop state feedback. This means that we want to substitute our input with an input that is based on the state of the system, i.e. $u = Kx$ where $K \in \mathbb{R}^{m \times n}$. We will consider the system to be in a stable closed loop form if all eigenvalues of $A + BK$ are stable (inside the unit circle). We can extend this notion to data-driven control as follows.

Def: Informative for state feedback [waarde2019data]
We say that the data (U_-, X) is informative for stabilization by state feedback if there exists a feedback gain K such that all systems described by the data are part of the set of systems that are stabilised using the gain K. I.e. $\Sigma_{i/s} \subseteq \Sigma_K$.

We will start by noting that if the data is informative for controllability or stabilisability then there is no guarantee that the there also exists a state feedback controller that stabilises all of the systems at once. However, if we find such a controller that we do know that all the systems described by the data are stabilisable. We will see an example of this later on in this section.

8.1 Mathematics

We will start by looking at the following theorem that gives us necessary and sufficient conditions for informativity for stabilisation by state feedback.

Thr: [waarde2019data]
The data (U_-, X) is informative for stabilisation by state feedback if and only if the matrix X_- has full row rank and there exists a right inverse $X_-^\dagger$ of X_- such that $X_+X_-^\dagger$ is stable. Moreover, K is such that $\Sigma_{i/s} \subseteq \Sigma_K$ if and only if $K = U_-X_-^\dagger$, where $X_-^\dagger$ is as described above.

From this theorem we can see that the problem can be reduced to finding a specific right inverse such that the systems are stable. From this right inverse we are able to construct the corresponding controller for stabilisation by state feedback.

Before we take a look at the proof of the theorem we will first note a Lemma that will be used in the proof.

Lem: [waarde2019data]
Suppose that the data (U_-, X) are informative for stabilisation by state feedback, and let K be a feedback gain such that $\Sigma_{i/s} \subseteq \Sigma_K$. Then $A_0 + B_0K = 0$ for all $(A_0, B_0) \in \Sigma^0_{i/o}$. Equivalently:

$$im \begin{bmatrix} I \\ K \end{bmatrix} \subseteq im \begin{bmatrix} X_- \\ U_- \end{bmatrix} \tag{6}$$

Proof '$\Leftarrow$':
Suppose X_- has full row rank and that there exists a right inverse $X_-^\dagger$ such that $X_+X_-^\dagger$ is stable. Let us define K as $K = U_-X_-^\dagger$. Let us consider the following:

$$X_+X_-^\dagger = \begin{bmatrix} A & B \end{bmatrix} \begin{bmatrix} X_- & U_- \end{bmatrix} X_-^\dagger = A + BK \tag{7}$$

This statement hold for all systems describing the data, $(A, B) \in \Sigma_{i/s}$. Hence we can conclude that K is a stabilising controller for all closed loop systems $A + BK$ with $(A, B) \in \Sigma_{i/s}$. Thus $\Sigma_{i/s} \subseteq \Sigma_K$. Hence we can conclude that the data is informative for stabilisation by state feedback.

Proof '⇒':
Suppose that the data is informative for stabilisation by state feedback. Let K be such that $A+BK$ is stable for all $(A, B) \in \Sigma_{i/s}$. By [**waarde2019data**] we know (6). This implies that X_- has full row rank and there exists a right inverse $X_-^\dagger$ such that:

$$\begin{bmatrix} I \\ K \end{bmatrix} = \begin{bmatrix} X_- \\ U_- \end{bmatrix} X_-^\dagger$$

By (7), we find that $A+BK = X_+X_-^\dagger$, which shows that $X_+X_-^\dagger$ is stable. Note that the stabilising feedback gain is given by $K = U_-X_-^\dagger$ concluding the proof.

However, in its current form it is not strait forward to compute a right inverse such that $X_+X_-^\dagger$ is stable. Hence we will use a different version of this theorem that has been rewritten in terms of linear matrix inequalities.

Thr: [waarde2019data]
The data (U_-, X) is informative for stabilisation by state feedback if and only if there exists a matrix $\Theta \in \mathbb{R}^{T \times n}$ satisfying

$$X_-\Theta = (X_-\Theta)^\top \qquad \begin{bmatrix} X_-\Theta & X_+\Theta \\ \Theta^\top X_+^\top & X_-\Theta \end{bmatrix} > 0$$

Moreover, K satisfies $\Sigma_{i/s} \subseteq \Sigma_K$ if and only if $K = U_-\Theta(X_-\Theta)^-1$ for some matrix Θ satisfying the above conditions.

This version of the theorem can be implemented using linear matrix inequality solvers such as Yalmip of CVX. The function provided in the toolbox are implemented using Yalmip.

8.2 Examples using implementation

The algorithm above is implemented in the following functions:

Syntax

```
[bool, K, diagnostics] = isInformStateFeedback(X, U)
[bool, K, diagnostics] = isInformStateFeedback(X, U, tolerance)
[bool, K, diagnostics] = isInformStateFeedback(X, U, tolerance, options)
```

Description

`[bool, K, diagnostics] = isInformStateFeedback(X, U)`: Returns if the data is informative for stabilisation by state feedback. If so, it also returns a corresponding controller K for closed loop feedback control of the form `A+BK`.
`[bool, K, diagnostics] = isInformStateFeedback(X, U, tolerance)`: Returns if the data is informative for stabilisation by state feedback given a specific tolerance. If so, it also returns a

corresponding controller K for closed loop feedback control of the form `A+BK`.
`[bool, K, diagnostics] = isInformStateFeedback(X, U, tolerance, options)`: Returns if the data is informative for stabilisation by state feedback given a specific tolerance and a spdsettings object. If so, it also returns a corresponding controller K for closed loop feedback control of the form `A+BK`.

Input arguments

`X`: State data matrix of dimension $n \times T + 1$.
`U`: Input data matrix of dimension $m \times T$.
`tolerance`: Tolerance used for determining when a value is zero up to machine precision. Default value is `1e-14`.
`options`: sdpsettings used with the Yalmip solver.

Output arguments

`bool`: (boolean) True if the data is informative for stabilisation by state feedback, false otherwise
`K`: (matrix) If the data is informative, it contains a stabilising controller `K` for closed loop control `A+BK`, empty otherwise.
`diagnostics`: (struct) Diagnostics from the Yalmip `optimize()` function.

8.2.1 Examples

9 Deadbeat control

In this section we will see how we can find a deadbeat controller using only our data. We will first discuss the underlying mathematics after which we will focus on how we can implement this into code. We will also look at a few examples on how to use the provided function.

Def: Informative for deadbeat control [waarde2019data]
We say the data (U_-, X) is informative for deadbeat control if there exists a feedback gain K such that $\Sigma_{i/s} \subseteq \Sigma_K^{nil}$.

In other words, the data is informative if we can stabilise all systems described by the data with a given K such that all closed loop systems only have the zero eigenvalue.

9.1 Mathematics deadbeat control

We will begin by considering the following theorem that gives necessary and sufficient conditions for informativity for deadbeat control.

Thr: [waarde20]
The data (U_-, X) is informative for deadbeat control if and only if the matrix X_- has full row rank and there exists a right inverse $X_-^\dagger$ of X_- such that $X_+X_-^\dagger$ is nilpotent. Moreover, if this condition is satisfied then the feedback gain $K := U_-X_-^\dagger$ yields a deadbeat controller.

Using this theorem the problem boils down to finding a suitable right inverse such that $X_-X_-^\dagger$ is nilpotent given that X_- has full row rank. We will consider the following 2 cases, first we will look at the case where X_- is a (full rank) square matrix. Second we will look at the case where X_- is a full row rank wide matrix.

First lets assume X_- is a square matrix and has full row rank. Then we know that the only right inverse of X_- is its inverse X_-^{-1}. Hence the data is informative for deadbeat control if and only if $X_+X_-^{-1}$ is nilpotent.

Now lets assume that X_- is a wide matrix and has full row rank. Then there exists an $F \in \mathbb{R}^{T\times n}$ spanning the row space of X_- and an $G \in \mathbb{R}^{T\times(T-n)}$ spanning the null space of X_- such that $\begin{bmatrix} F & G \end{bmatrix}$ is non singular and $X_- \begin{bmatrix} F & G \end{bmatrix} = \begin{bmatrix} I_n & 0_{n\times(T-n)} \end{bmatrix}$. Now let $X_-^\dagger = F + GH$ where $H \in \mathbb{R}^{(T-n)\times n}$ if and only if $X_-^\dagger$ is an right inverse of X_-.

Proof '$\Rightarrow$':
Let $X_-^\dagger = F + GH$ as described above. Then $X_-X_-^\dagger = X_-F + X_-GH = I_n + 0_{n\times(T-n)} * H = I_n$.

Proof '$\Leftarrow$':

add proof

Thus finding a right inverse such that $X_+X_-^\dagger$ is nilpotent is equivalent to finding an H such that $X_+F + X_+GH$ is nilpotent. Finding this H can be done using pole placement methods for the pair (X_+F, X_+G) and placing all eigenvalues at zero.

However, due to limitation in Matlab's pole placement function we are not able to implement the algorithm into Matlab directly. This is because Matlab has 2 functions for pole placement, the `place()` function which does not support poles with high multiplicity, which is hence not useable. The other function `acker()` does support pole placement of poles with high multiplicity, but it is limited to a single dimensional input space. Since X_+G is an $n \times (T-n)$ matrix, we have that our input space is of dimension $T-n$. However, we are able to circumvent this issue by extending the `acker()` function to support higher dimensional input. To do this we will use the proof of the sufficiency in theorem 3.29 as well as lemma 3.31 from [**bookTrentelman**].

9.2 Mathematics extending pole placement

Lem: [bookTrentelman]
If (A, B) is controllable there exist vectors $u_0, \dots u_{n-1}$ such that the vectors defined by

$$x_0 := 0, \;\; x_{k+1} := Ax_k + Bu_k \;\; (k = 0, \dots, n-1)$$

are independent.

Assume that the pair (A, B) is controllable, we will use [**bookTrentelman**] to construct $u_0, \dots, u_{n-1}$ and $x_0, x_1, \dots, x_n$. We pick u_n to be arbitrary. Then there exists a unique map F_0 such that $F_0 x_k = u_k$ for $k = 1, \dots, n$. We can substitute this to get:

$$x_{k+1} = Ax_k + BF_0 x_k = (A + BF_0)x_k$$

for $k \in [1, \dots, n]$. From this we can see that $x_k = (A + BF_0)^{k-1} x_1$. By construction of lemma 3.31 we have that $x_0 = 0$, thus we have $x_1 = A * 0 + B * u_0 = Bu_0$. We will call this $b = Bu_0 = x_1$. By construction $x_1, \dots, x_n$ are independent. Thus we know that the controllability matrix $\begin{bmatrix} b & (A+BF_0)b & \dots & (A+BF_0)^{n-1}b \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}$ is full rank. Hence the pair $(A+BF_0, b)$ is controllable. Since the matrix b is a column vector we can use the `acker()` algorithm to compute a suitable controller f such that $A + BF_0 + bf = A + B(F_0 + u_0 f)$ has appropriate eigenvalues. From this we construct the controller $F = F_0 + u_0 f$ that places the eigenvalues in the chosen positions.

9.3 Examples using implementation

The algorithm above is implemented in the following functions:

Syntax

```
[bool, K] = isInformDeadbeatControl(X, U)
```

Description

`[bool, K] = isInformDeadbeatControl(X, U)`: Returns if the data is informative for deadbeat control. If so, it also returns a corresponding controller K for closed loop feedback control of the form `A+BK`.

Input arguments

`X`: State data matrix of dimension $n \times T + 1$.
`U`: Input data matrix of dimension $m \times T$.

Output arguments

`bool`: (boolean) True if the data is informative for deadbeat control, false otherwise
`K`: (matrix) If the data is informative, it contains a stabilising controller `K` for closed loop control `A+BK`, empty otherwise.

9.3.1 Examples

In this example we will consider the following data:

$$X = \begin{bmatrix} 0 & 2 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} \qquad U = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

Before we look if the data is informative for deadbeat control, we will fist consider if the data is informative for system identification and controllability. Recall that for system identification we need that $\begin{bmatrix} X_- \\ U_- \end{bmatrix}$ is full row rank and for controllability we need that $X_+ - \lambda X_-$ is full row rank for all $\lambda \in \mathbb{C}$.

$$rank(\begin{bmatrix} X_- \\ U_- \end{bmatrix}) = n + m \qquad rank(X_+ - \lambda X_-) = n$$

$$rank(\begin{bmatrix} 0 & 2 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}) = 3 \qquad rank(\begin{bmatrix} 2 & 1-2\lambda & -\lambda \\ 1-\lambda & -\lambda & 0 \end{bmatrix}) = 2$$

Hence the data is informative for system identification and controllability. Due to the being able to identify the system and the system being controllable we expect that we are able to find a deadbeat controller. We will first attempt to construct one from the data directly. For this we will first need to find an F and G such that $X_- \begin{bmatrix} F & G \end{bmatrix} = \begin{bmatrix} I_2 & 0_{2\times 1} \end{bmatrix}$. We find the following F and G:

$$F = \begin{bmatrix} -\frac{1}{3} & \frac{5}{6} \\ \frac{1}{3} & \frac{1}{6} \\ \frac{1}{3} & -\frac{1}{3} \end{bmatrix} \qquad G = \begin{bmatrix} \frac{1}{\sqrt{6}} \\ -\frac{1}{\sqrt{6}} \\ \frac{\sqrt{2}}{\sqrt{3}} \end{bmatrix}$$

Any right inverse of X_- can be expressed as $X_-^\dagger = F + GH$ where $H \in \mathbb{R}^{1\times 2}$. Recall that we need to find a right inverse such that $X_+X_-^\dagger$ is nilpotent. Hence we need to find an H such that $X_+F + X_+GH$ only has zero eigenvalues.

$$X_+F + X_+GH = \begin{bmatrix} -\frac{1}{3} & \frac{11}{6} \\ -\frac{1}{3} & \frac{5}{6} \end{bmatrix} + \begin{bmatrix} \frac{1}{\sqrt{6}} \\ \frac{1}{\sqrt{6}} \end{bmatrix} \begin{bmatrix} h_1 & h_2 \end{bmatrix}$$

By applying pole placement methods we can find that $H = \begin{bmatrix} \frac{\sqrt{2}}{\sqrt{3}} & \frac{-5}{\sqrt{6}} \end{bmatrix}$ does indeed provide a right inverse such that $X_+X_-^\dagger$ is nilpotent. Using this right inverse we can construct the deadbeat controller K as follows.

$$K = U_- * X_-^\dagger = U_-(F + GH) = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 0 & 0 \end{bmatrix}$$

At this point we will recall that we where able to identify the system using the data. If we do this we find that the system is given by:

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

As we can see the system already has only zero eigenvalues, hence it will follow the properties of deadbeat control if we provide no input.

We can also find the same results by using the Matlab function:

```
X = [0 2 1 0 ; 1 1 0 0]; U = [1 0 0];
[bool, K] = isInformDeadbeatControl(X,U)
```

Which will return: `[ 1, [0.5329e-14 -0.6217e-14] ]`.

10 Linear quadratic regulation

In this section we will see how we can construct a linear quadratic regulator (LQR) based on a given cost function directly from the input-state data. We will first define what LQR control entails for normal control theory after which we will consider its data-driven dual. We will also see how this data-driven version can be applied to retrieve the controller from the data. Lastly we will see how to solve such a problem both by doing it by hand as well as using the provided Matlab functions.

10.1 Non data-driven LQR

We will first revisit the topic of non data-driven LQR. To given an intuitive idea, we want to not only find a solution to the stabilisation problem, we also want to find the best solution given some cost function. Let us consider the discrete-time linear systems of the form $x(t+1) = Ax(t) + Bu(t)$. We will denote the state sequence generated by this system based of a given initial condition x_0 and input function u as $x_{x_0,u}(\cdot)$. We will also define a quadratic cost function associated to the system.

$$J(x_0, u) = \sum_{t=0}^{\infty} x^\top(t)Qx(t) + u^\top(t)Ru(t)$$

Where Q is symmetric positive semi definite and R is symmetric positive definite. Using these notions want to find for every initial condition x_0 an input $\bar{u}$ such that the system stabilises and the cost function is minimised as well as finite, In other words, the $\lim_{t\to\infty} x_{x_0,\bar{u}}(t) = 0$ given that the cost function $J(x_0, \bar{u}) \leq J(x_0, u)$ for all inputs u that cause the system to stabilise. Such an input $\bar{u}$ is called optimal for the given initial condition. As we might expect, an optimal solution does not always exist for every initial condition. Hence we say that an LQR problem is solvable for given (A, B, Q, R) if for every initial condition x_0 there exists an optimal solution.

Using the following theorem we can find the solution to the LQR problem. Recall from [**bookTrentelman**] that an eigenvalue is called (Q, A)-observable if

$$rank \begin{pmatrix} A - \lambda I \\ Q \end{pmatrix} = n$$

where n is the dimension of the state space.

Thr: [waarde2019data]
let Q be symmetric positive semi definite and R be symmetric positive definite. Then the following statements hold:

1. *If (A, B) is stabilisable, there exists a unique largest real symmetric solution P^+ to the discrete-time algebraic Riccati equation*

$$P = A^\top PA - A^\top PB(R + B^\top PB)^{-1}B^\top PA + Q \tag{8}$$

in the sense that $P^+ \geq P$ for every real symmetric P satisfying the discrete-time algebraic Riccati equation. The matrix P^+ is positive semi definite.

2. *If, in addition to stabilisability of (A, B), every eigenvalue of A on the unit circle is (Q, A)-observable then for every x_0 a unique optimal input $\bar{u}$ exists. Furthermore, this input sequence*

is generated by the feedback law $u = Kx$, where K is given by:

$$K := -(R + B^\top P^+ B)^{-1} B^\top P^+ A \tag{9}$$

Moreover, the matrix $A + BK$ is stable.

3. *In fact, the linear quadratic regulator problem is solvable for (A, B, Q, R) if and only if (A, B) is stabilisable and every eigenvalue of A on the unit circle is (Q, A)-observable.*

If a LQR problem is solvable then we call K the optimal feedback gain for (A, B, Q, R).

10.2 Data-driven LQR

Now that we have familiarised ourselves with the concept of LQR control we will see how to extend it to data-driven control. We will start by defining the set of systems for which a solution is valid given a feedback gain K and cost matrices Q and R.

$$\Sigma_K^{Q,R} := \{(A, B) \,:\, K \text{ is the optimal gain for } (A, B, Q, R)\}$$

Using this we can define informativity for linear quadratic regulation as follows.

Def: Informative for linear quadratic regulation [waarde2019data]
Given matrices Q and R, we say that the data (U_-, X) is informative for linear quadratic regulation if there exists feedback gain K such that the optimal gain for all systems is K, i.e. $\Sigma_{i/s} \subseteq \Sigma_K^{Q,R}$.

We will now consider the following theorem that gives us necessary and sufficient conditions for finding the solution of a data-driven LQR problem.

Thr: [waarde2019data]
Let Q be symmetric positive semi definite and R be symmetric positive definite. Then the data (U_-, X) is informative for linear quadratic regulation if and only if at least one of the following two conditions hold:

1. *The data (U_-, X) is informative for system identification, that is $\Sigma_{i/s} = \{(A_s, B_s)\}$, and the linear quadratic regulator problem is solvable for (A_s, B_s, Q, R). In this case, the optimal feedback gain K is of the form (9) where P^+ is of the form (8).*

2. *For all $(A, B) \in \Sigma_{i/s}$ we have $A = A_s$. Moreover, A_s is stable, $QA_s = 0$, and the optimal feedback gain is given by $K = 0$.*

As we can see from the theorem if we are able to identify the system then the data-driven LQR problem reduces to a normal LQR problem for which we have already seen how to solve them. However, there is still the case in which all systems have the same $A = A_s$ matrix, A_s matrix is inherently stable and $QA_s = 0$. We will now look at why this is a valid solution to the LQR problem in the first place.

Assume we have a A_s which is stable and a Q such that $QA_s = 0$. From this point we assume B to be an arbitrary input matrix of dimension $n \times m$. Since $x(t) \in im(A)$ for all $t > 0$ we have that $Qx(t) = 0$ for all $t > 0$ if we pick our input function to be zero. Note that if we pick our input to be zero then the input cost of the cost function will also be zero. Hence the cost will be equal to $J(x_0, u) = x_0^\top Q x_0$, which is finite. Since the system is inherently stable we know that the system will stabilise without any input. Lastly we note that the cost associated with the initial condition is present in all cost sequences generated for different inputs, hence the zero input is the minimal solution to the cost function. Hence this case will provide a valid solution to our LQR problem.

However, even though it is a valid solution it is not easily computable in its current form. Hence we will consider the following theorem which rewrites the condition to one of linear matrix inequalities which can be solved by computational means.

Thr: [waarde2019data]
Let Q be symmetric positive semi definite and R be symmetric positive definite. Then the data (U_-, X) is informative for linear quadratic regulation if and only if at least one of the following two conditions hold:

1. *The data (U_-, X) is informative for system identification, that is $\Sigma_{i/s} = \{(A_s, B_s)\}$, and the linear quadratic regulator problem is solvable for (A_s, B_s, Q, R). In this case, the optimal feedback gain K is of the form (9) where P^+ is of the form (8).*
2. *There exists $\Theta \in \mathbb{R}^{T\times n}$ such that $X_-\Theta = (X_-\Theta)^\top$, $U_-\Theta = 0$, $QX_+\Theta = 0$ and*
$$\begin{bmatrix} X_-\Theta & X_+\Theta \\ \Theta^\top X_+^\top & X_-\Theta \end{bmatrix}$$

10.3 Examples using implementation

The algorithm above is implemented in the following functions:

Syntax

```
[bool, K, diagnostics] = isInformLQR(X, U, Q, R)
[bool, K, diagnostics] = isInformLQR(X, U, Q, R, tolerance)
[bool, K, diagnostics] = isInformLQR(X, U, Q, R, tolerance, options)
```

Description

`[bool, K, diagnostics] = isInformLQR(X, U, Q, R)`: Returns if the data is informative for LQR control. If so, it also returns a corresponding controller `K` for the closed loop feedback control of the form `A+BK`.
`[bool, K, diagnostics] = isInformLQR(X, U, Q, R, tolerance)`: Returns if the data is informative for LQR control given a specific tolerance. If so, it also returns a corresponding controller `K` for the closed loop feedback control of the form `A+BK`.
`[bool, K, diagnostics] = isInformLQR(X, U, Q, R, tolerance, options)`: Returns if the data is informative for LQR control given a specific tolerance and sdpsettings. If so, it also returns a corresponding controller `K` for the closed loop feedback control of the form `A+BK`.

Input arguments

`X`: State data matrix of dimension $n \times T+1$.
`U`: Input data matrix of dimension $m \times T$.

`Q`: State cost matrix.
`R`: Input cost matrix.
`tolerance`: Tolerance used for determining when a value is zero up to machine precision. Default value is `1e-14`.
`options`: sdpsettings used with the Yalmip solver.

Output arguments

`bool`: (boolean) True if the data is informative for system identification, false otherwise
`K`: (matrix) If the data is informative, it contains a stabilising controller `K` for closed loop control `A+BK`, empty otherwise.
`diagnostics`: (struct) Diagnostics from the Yalmip `optimize()` function.

10.3.1 Examples