Technical

MANUAL

Tensegrity Data-based Modeling and Control (TsgDMC) Software

@ College of Future Science and Engineering, Suzhou University, China
@ Department of Mechanical and Aerospace Engineering, University of Kentucky, USA
@ Department of Aerospace Engineering, Texas A&M University, USA

September 2023

Revision Sheet

Release No.	**Date**	**Revision Description**
Rev. 0.0	10/2022	Developed functions for the Markov Data-based Control Algorithm
Rev. 0.1	04/2023	Revised Regulating Control, Reference Tracking codes
Rev. 0.2	05/2023	Revised ERA, QMC codes
Rev. 0.3	06/2023	User's Manual Created
Rev. 0.4	08/2023	Updated User's Manual
Version 1.1	09/2023	Submitted to JOSS (Journal of Open Source Software)

Tensegrity Data-based Modeling and Control (TsgDMC)

Software Information

Software Goals:

This software aims to facilitate the database modeling and control of tensegrity structures. The software offers two data-based modeling methods: the Eigensystem Realization Algorithm (ERA) and the q-Markov Covariance Equivalent Realization (QMC). Additionally, it provides a data-based control technique to implement a Linear Quadratic Gaussian (LQG) control law for both regulation and reference tracking control.

- **Data-based Modeling**
  1). Eigensystem Realization Algorithm
  2). Q-Markov Covariance Equivalent Realization
- **Data-Based Control**:
  1). Regulatory Control
  2). Reference Tracking

TsgDMC Members and Information:

Yuling Shen
ylshen07@suda.edu.cn, Ph.D., Associate Professor, School of Future Science and Engineering, Soochow University, Suzhou, Jiangsu, 215222, China

Muhao Chen
muhaochen@uky.edu, Ph.D., Assistant Professor, Department of Mechanical and Aerospace Engineering, University of Kentucky, Lexington, Kentucky, USA

Robert E. Skelton
bobskelton@tamu.edu, Ph.D., TEES Eminent Professor, Member National Academy of Engineering, Department of Aerospace Engineering, Joint Faculty in Department of Ocean Engineering, Texas A&M University, College Station, Texas, USA

TABLE OF CONTENTS

1.0 GENERAL INFORMATION

1. GENERAL INFORMATION

1.1 System Overview

Undergraduate linear algebra, material mechanics/continuum mechanics, finite element method, linear control theory, and some basic knowledge of MATLAB are required to understand the codes well. This software is developed based on the following:

- 64-bit Windows
- MATLAB

Note: Win7/Win10/Mac OS/Linux/Win XP/Win Vista, the software is compatible with a MATLAB version later than 2009a. However, if possible, we encourage users to run the software with the latest MATLAB release. (More information about MATLAB versions can be found here: https://en.wikipedia.org/wiki/MATLAB).

1.2 Project References

Tensegrity Data-based Modeling and Control (TsgDMC) software is created based on the theories developed in the following references.

[1] Shen, Y., Chen, M., Skelton, R.E., *A Markov data-based approach to system identification and output error covariance analysis for tensegrity structures*. Composites: Parts B. Under review.
[2] Shen, Y., Chen, M., Majji, M., Skelton, R.E., *An Efficient q-Markov Covariance Equivalent Realization Approach to System Identification*. In IFAC World Congress 2023.
[3] Shen, Y., Chen, M., Majji, M. and Skelton, R.E., 2023. *Q-Markov Covariance equivalent realizations for unstable and marginally stable systems*. Mechanical Systems and Signal Processing, 196, p.110343. Doi: https://doi.org/10.1016/j.ymssp.2023.110343
[4] Ma, S., Chen, M., Skelton, R.E., 2022. *Tensegrity system dynamics based on finite element method*. Composite Structures 280, 114838. Doi: http://dx.doi.org/https://doi.org/10.1016/j.compstruct.2021.114838
[5] Ma, S., Chen, M., Skelton, R.E., 2022. *Tsgfem: Tensegrity finite element method*. Journal of Open Source Software 7, 3390. Doi: http://dx.doi.org/https://doi.org/10.21105/joss.03390
[6] Shen, Y., Chen, M., Skelton, R.E., *Markov data-based reference tracking control to tensegrity morphing airfoils*. Engineering Structures. 291, 116430. Doi: https://doi.org/10.1016/j.engstruct.2023.116430

1.3 Authorized Use Permission

1.4 Points of Contact

1.4.1 Information

Our group focuses on the research of integrating structure and control design. Based on the tensegrity paradigm, we design tensegrity structures to meet the specified objectives. These objectives can vary from minimizing the structure's mass to controlling the structure to meet specific performances. This software is intended to study the data-based modeling and control of tensegrity structures. The authors would like to make this open-source software to help other researchers interested in this field.

This user guide states every aspect of the software to make it more user-friendly. We appreciate your questions and any help in improving the software.

1.4.2 Help Desk

We are open and willing to answer any question. Please state your problem clearly and use the following emails to contact: **Muhao Chen**: muhaochen@uky.edu, **Yuling Shen**: ylshen07@suda.edu.cn.

1.5 Organization of the Manual

User's Manual v1.1.

2.0 SYSTEM SUMMARY

2. SYSTEM SUMMARY

2.1 System Configuration

This software does not have a specific APP user interface; a MATLAB .mat file is implemented as one. The user should make sure MATLAB (https://www.mathworks.com/products/matlab.html) is well installed. Following the steps below, one can perform the statics analysis and dynamics simulations for any tensegrity structure.

2.2 Data-based Modeling and Control Steps

To identify or control a system, the user should follow these steps (more details will be provided in the following chapters):

2.2.1 Data-based Modeling

- **Specify System Inputs and Outputs**: Determine the input and output signals assigned for the system.
- **Evaluate Parameters:** Compute Markov/Covariance parameters using Stochastic methods from I/O tests or deterministic methods from model parameters.
- **Construct Data Matrix:** For ERA, determine Hankel matrix size and construct two sequential Hankel matrices using Markov parameters; for QMC, specify the number of the Markov and Covariance parameters for the identification system to match exactly, then construct the data matrix according to the exact match number using Markov and Covariance parameters.
- **Determine Model Complexity:** Determine the reduced order to preserve from the singular value decomposition of the Hankel matrix or data matrix.
- **Compute Solution:** Calculate the approximation model parameters of the reduced order using ERA or QMC.
- **Results Analysis:** Perform system simulations to verify the identified system.
- **Exit System:** Click on Exit to close MATLAB.

2.2.2 Data-based Control

- **Specify System Inputs and Outputs**: Determine the input and output signals assigned for the system. Specify the reference trajectories for the outputs to track in RTC.
- **Evaluate Parameters:** Compute Markov parameters using Stochastic methods from I/O tests or deterministic methods from model parameters. Transform the Markov parameters into the augmented Markov parameters for RTC.
- **Specify Horizon and Weights:** Specify the control horizon and the weight matrices for inputs and outputs.
- **Compute Control Sequence:** Compute the corresponding control input sequence, which minimizes the cost function over the pre-specified horizon and weight matrices.
- **Results Analysis**: Perform system simulations to verify the control performance.
- **Make Video**: Make a video of the control process in every sub-step.
- **Exit System**: Click on Exit to close MATLAB.

2.3 Become a Developer

We strongly recommend users delve into the papers detailed in Section 1.2 to deeply comprehend the software codes and expand its functionalities for specialized tasks. We welcome collaborations on tensegrity research endeavors and value your insights on both theoretical and hands-on challenges.

3.0 INSTRUCTION FOR DATA-BASED MODELING

3. INSTRUCTION FOR DATA-BASED MODELING

This section provides detailed procedures for performing system identification using the Eigensystem Realization Algorithm (ERA) or q-Markov Covariance Equivalent Realization (QMC), accompanied by several example scripts that serve as illustrations. These example scripts have been carefully crafted to highlight different features and provide a practical understanding of the software's capabilities.

3.1 Evaluate Markov/Covariance Parameters

System identification techniques require processed parameters from I/O data. Specifically, ERA requires Markov parameters, and the QMC requires both Markov and Covariance parameters to proceed. Both parameters can be acquired via either I/O tests or Model parameters. In the application of black box system identification, we use I/O tests to evaluate these parameters. A sample code that computes the first 1000 Markov parameters from I/O tests is attached below:

```
%% Generate Markov parameters

dt = 0.01; % Define time step
N = 1000;  % Number of parameters to evaluate
func = @simple_pendulum; % Define black box system
us = ones(N,1);
xs = zeros(2,1);
for i = 1:N % I/O test
    [~,xs_temp] = ode45(func,[0,dt],xs,[],us(i));
    xs = xs_temp(end,:)';
    ys_store(:,i+1) = xs(1);
end

for i = 1:length(ys_store) % Data process
    if i == 1
        H(:,1,i) = ys_store(:,i);
    else
        H(:,1,i) = (ys_store(:,i)-ys_store(:,i-1));
    end
end
```

In the application of model reduction, model parameters of the high-dimensional system are usually available. The computation of Markov and Covariance parameters from model parameters is facilitated with the functions 'tsgDMC_genMarkov' and 'tsgDMC_genCov.' A sample code for running the two functions to compute the first 100 Markov and Covariance parameters is given below:

```
%% Generate Markov and Covariance Parameters

H = tsgDMC_genMarkov(A,B,C,D,100);
R = tsgDMC_genCov(A,B,C,D,100);
```

3.2 Construct Data Matrix

Data matrices shall be constructed in the next step. In ERA, two Hankel matrices of self-defined sizes shall be constructed. This is facilitated by the function 'tsgDMC_Hankel.' A sample code that computes the first two Hankel matrices with 400 Markov parameters in rows and columns is attached below:

```
%% Construct Hankel matrices

a = 20; % Specify number of Markov parameters in the row
b = 20; % Specify number of Markov parameters in the column
[Phi1,Phi2] = tsgDMC_Hankel(H,a,b);
```

In QMC, two Toeplitz matrices consisting of exactly matched Markov and Covariance parameters shall be formulated first, and a data matrix is then calculated. An existence condition shall also be checked for the data matrix to see if a QMC solution exists. These steps are facilitated in functions 'tsgDMC_Hq,' 'tsgDMC_Rq,' 'tsgDMC_Dq,' and 'tsgDMC_existence.' A sample code that illustrates the above procedure, which computes the data matrix when $q = 10$ and, in addition, checks the existence condition, is attached below:

```
%% Construct data matrix for QMC

q = 10; % Specify the number of parameters to exactly match
Hq = tsgDMC_Hq(H,q); % Construct Toeplitz matrix of Markov parameters
Rq = tsgDMC_Rq(R,q); % Construct Toeplitz matrix of Covariance parameters
Data_q = tsgDMC_Dq(Hq,Rq); % Construct data matrix
existence = tsgDMC_existence(Data_q); % Check if a QMC solution exists
```

3.3 Determine Model Complexity

The Markov and Covariance parameters may consist of noise. In most conditions, it is possible to represent most information in the data matrix using a reduced-order matrix. The singular value decomposition of the data matrices shall be performed, and the preserved power density of the singular values can determine the reduced order. The corresponding function for this step is 'tsgDMC_pdm'. A sample code that finds the model complexity, which preserves 99.9999% power density of the data matrix (first Hankel matrix for ERA, and data matrix for QMC), is attached below:

```
%% Determine model complexity for ERA

sigma = 0.999999; % Specify the power density to perserve
n = tsgDMC_pdm(Phi1,sigma);

%% Determine model complexity for QMC

sigma = 0.999999; % Specify the power density to perserve
n = tsgDMC_pdm(Data_q,sigma);
```

3.4 Compute Identification Model Parameters

This step computes the reduced-order state space realization using either ERA or QMC method from the computed parameters. They are facilitated as functions 'tsgDMC_era' and 'tsgDMC_qmc.' The sample code that finds an ERA or QMC solution of state size n is attached below:

```
%% Compute Identification Model Parameters using ERA

[A,B,C,D] = tsgMDC_era(Phi1,Phi2,H,n);
```

```
%% Determine model complexity for QMC

sigma = 0.999999; % Specify the power density to perserve
n = tsgDMC_pdm(Data_q,sigma);
```

4.0 INSTRUCTIONS FOR DATA-BASED CONTROL

4. INSTRUCTION FOR DATA-BASED CONTROL

4.1 Evaluate Markov Parameters

Markov parameters of the system are required to compute the control sequence. They can be acquired via either I/O tests or Model parameters. The sample code for an I/O test method has been attached below:

```
%% Generate Markov parameters

dt = 0.01; % Define time step
N = 1000;  % Number of parameters to evaluate
func = @simple_pendulum; % Define black box system
us = ones(N,1);
xs = zeros(2,1);
for i = 1:N % I/O test
    [~,xs_temp] = ode45(func,[0,dt],xs,[],us(i));
    xs = xs_temp(end,:)';
    ys_store(:,i+1) = xs(1);
end

for i = 1:length(ys_store) % Data process
    if i == 1
        H(:,1,i) = ys_store(:,i);
    else
        H(:,1,i) = (ys_store(:,i)-ys_store(:,i-1));
    end
end
```

For reference tracking control, the Markov parameters shall be transformed into augmented Markov parameters, which is facilitated in function 'tsgDMC_markov_transform' and is employed as the following:

```
%% Transform Markov Parameters

[H_hat,M_hat] = tsgDMC_markov_transform(H,M);
```

4.2 Specify Required Parameters

Several parameters shall be specified to proceed with the control. Specifically, N is the horizon length defined by the cost function. Q and R are weight matrices for output and input in regulation control and for tracking errors and input increments in reference tracking control. An additional reference trajectory r shall also be specified for reference track control. A sample code is attached below:

```
%% Specify Required Parameters

r = Simple_Pendulum.r; % Specify reference trajectory
N = size(r,2)-1; % Specify Horizon length
Q = 1e0*eye(size_output); % Tracking error weight matrix
R = 1e0*eye(size_input); % Input increment weight matrix
```

4.3 Compute Control Sequence

Within a control horizon N, the optimal controller gain K at current step k shall be computed using the function 'tsgDMC_gain.'

```
%% Compute Optimal Controller Gain

K = tsgDMC_gain(k,N,Q,R,H_hat);
```

4.4 Compute Output Estimate

Actuation noise variance W and sensor noise variance V shall be specified. Then, outputs at the current state k will be estimated using a Kalman filter, as implemented in the function 'tsgDMC_estimate.' A sample code of this procedure is attached below:

```
%% Compute Output Estimate

W = 1e-6*eye(size_input); % Specify actuator noise variance
V = 1e-6*eye(size_output); % Specify sensor noise variance
y_est = tsgDMC_estimate(k,N,y_est,y,u(1:size_input),H_hat,M_hat,W,V);
```

4.5 Compute input Sequence

The input (input increment) sequence that minimizes the cost function over the horizon N is computed using the function 'tsgDMC_control.' For regulation control, the optimal controller gain K and the current output estimate y_{est} are required to compute the input sequence as follows:

```
%% Compute Optimal Input Sequence for Regulation Control

u = tsgDMC_control(K,y_est); % Compute Input Sequence
```

For reference tracking control, the optimal controller gain K, current output estimate y_{est} and reference trajectory are required to compute the input increment sequence. The current input shall be processed as the sum of the last input and the current input increment. A sample code for this procedure is attached below:

```
%% Compute Optimal Input Increment Sequence for Reference Tracking Control

u_increment = tsgDMC_control(K,y_est,r); % Compute Input Increment Sequence
u_last = u; % Store last input signal
u = u_increment(1:size_input) + u_last; % Compute current input signal
```

5.0 APPENDIX

5. APPENDIX

5.1 Example Configuration

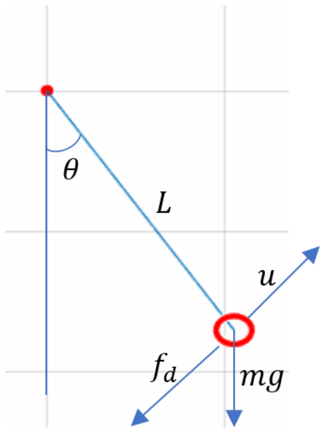

Figure 1 A Pendulum Example for Software Verification.

A nonlinear simple pendulum is studied as an example, shown in Figure 1. The dynamics of the pendulum are given as the following:

$$mL\ddot{\theta} + f_d + mgsin\theta = u,$$

where m represents the point mass, L is the bar length, f_d denotes the damping force assumed as $f_d = cL\dot{\theta}$, c is the damping ratio, θ gives the rotating angle, and u represents the input in force. The mass of the bar is assumed to be negligible.

The dynamics of the simple pendulum are packed as a black box system. The input is specified as the force u, and the output is specified as the rotating angle θ. The following properties have been selected for this study:

Table 1. The structure parameters of the single pendulum.

Parameters	*Values*	*Units*
m	1	kg
L	1	m
c	0.1	Ns/m
g	9.8	$\mathrm{m/s^2}$

5.2 Verification of System Identification

In this section, an approximation system of the simple pendulum system using ERA has been computed. Both systems are excited by a pulse, step, white noise, and sine inputs for 10 seconds to verify their responses, where the results are depicted in Figure 2. The result demonstrates the ERA has matched the nonlinear simple pendulum correctly, with a response discrepancy at a scale of 10^{-3} or less.

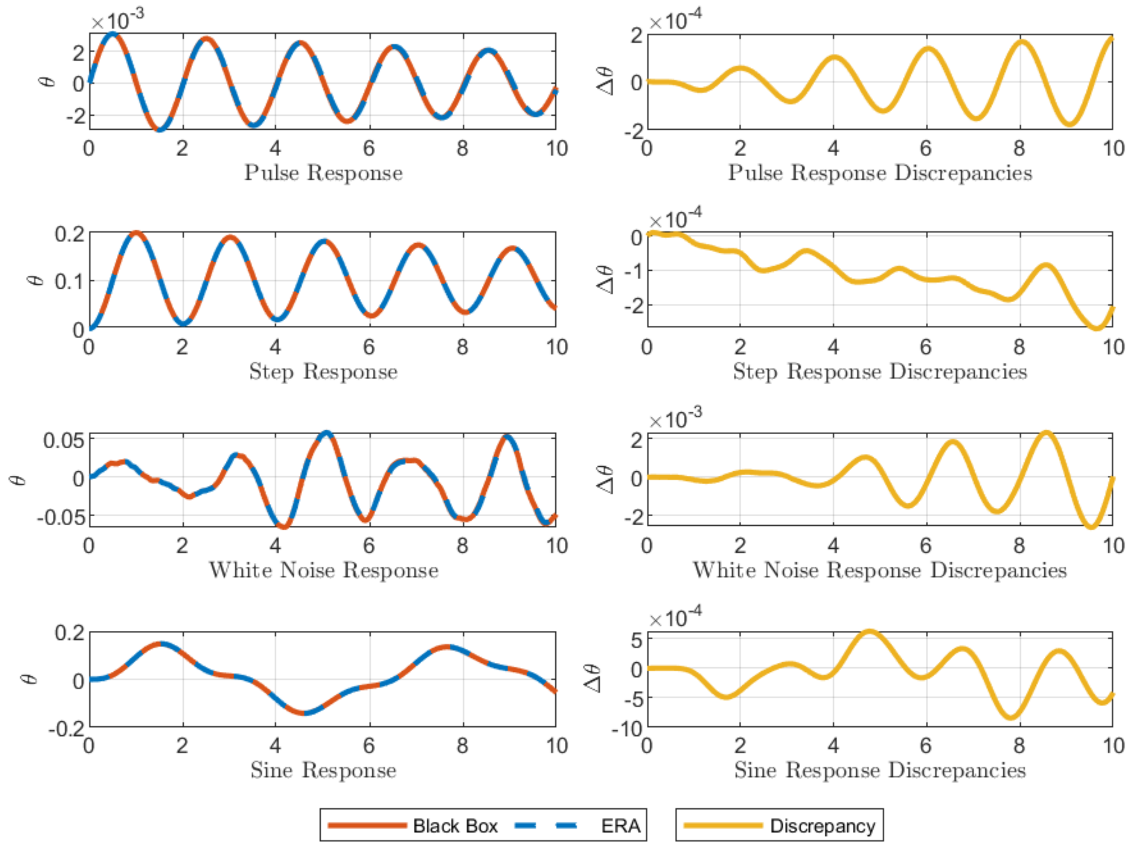

Figure 2 Responses from the black box and its associated ERA systems when stimulated by pulse, step, white noise, and sine inputs for 10 seconds, along with the discrepancy in their responses.

5.3 Verification of Markov Data-based Control

This section demonstrates the Markov data-based control of the pendulum at three different weights. The reference signal is set as $r = \frac{\pi}{6}$, where we want to drive the point mass to the position $\theta = \frac{\pi}{6}$ while minimizing the cost function consisting of an accumulation of tracking errors and input increments over the horizon $N = 50$. Three different weight combinations have been selected: (1) $Q = 1, R = 1$, the weight of tracking error is low, and the system control will not use small efforts to track the reference; (2) $Q = 1000, R = 1$, the weight of tracking error is medium thus the system control uses medium efforts; (3) $Q = 1E6, R = 1$, the weight of tracking error is high and the system control will use considerable efforts to track the reference. Figure 3 shows these three selections' tracking errors and input increments over the horizon. The result matches our prediction of the control process, thus demonstrating the effectiveness of the Markov data-based control.

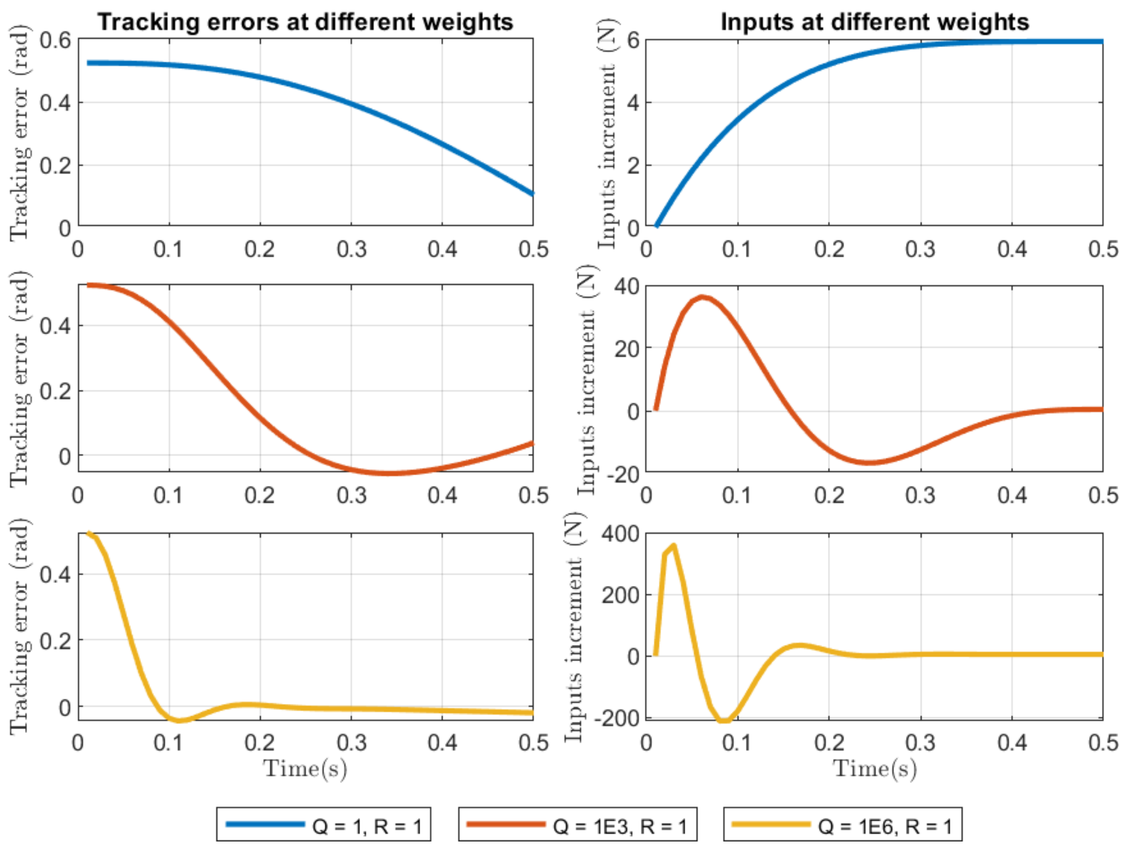

Figure 3: Tracking errors and input increments over the horizon for three chosen weight combinations: (1) Q=1, R=1, (2) Q=1000, R=1, and (3) Q=1E6, R=1.