

# Package ‘netcontrol’

October 13, 2022

**Type** Package

**Title** Control Theory Methods for Networks

**Version** 0.1

**Date** 2020-02-11

**Author** Teague R. Henry

**Maintainer** Teague R. Henry <trhenry@email.unc.edu>

**Description** Implementations of various control theory methods for use in brain and psychological networks. Contains controllability statistics from Pasqualetti, Zampieri & Bullo (2014) <[doi:10.1109/TCNS.2014.2310254](https://doi.org/10.1109/TCNS.2014.2310254)>, optimal control algorithms from Lewis, Vrabie & Syrmos (2012, ISBN:978-0-470-63349-6), and various utilities.

**License** MIT + file LICENSE

**Imports** Rcpp (>= 1.0.1), Rdpack, Matrix, MASS, pracma, expm

**LinkingTo** Rcpp, RcppArmadillo

**RoxygenNote** 7.0.2

**Encoding** UTF-8

**RdMacros** Rdpack

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2020-03-19 12:40:02 UTC

## R topics documented:

average_control . . . . .	2
ave_control_centrality . . . . .	3
control_gramian . . . . .	3
control_scheme_DLI_freestate . . . . .	4
control_traj . . . . .	5
dlyap . . . . .	7
inv_average_control . . . . .	8
LQR . . . . .	8

modal_control . . . . .	9
modal_control_centrality . . . . .	10
netcontrol . . . . .	11
<b>Index</b>	<b>12</b>

---

average_control	average_control - Average controllability as defined by the trace of the Gramian
-----------------	--

---

### Description

A commonly used measure (Pasqualetti et al. 2014) of the overall controllability of a system defined by  $x(t+1) = Ax(t) + Bu(t)$ .

### Usage

```
average_control(A, B)
```

### Arguments

A	A $n \times n$ matrix.
B	A $n \times m$ matrix.

### Value

Trace of the infinite time Gramian.

### References

Pasqualetti F, Zampieri S, Bullo F (2014). “Controllability Metrics, Limitations and Algorithms for Complex Networks.” In *2014 American Control Conference*, 3287–3292. ISBN 978-1-4799-3274-0 978-1-4799-3272-6 978-1-4799-3271-9, doi: [10/ggkhs9](https://doi.org/10/ggkhs9).

### Examples

```
A = matrix(c(0,-3,-2,2,-2,1,-1,2,-1), 3,3)
B = diag(3)

average_control(A, B)
```

---

ave\_control\_centrality  
*Average Control Centrality*

---

**Description**

Calculates the average control centrality of a system defined by  $x(t+1) = Ax(t) + Bu(t)$ .

**Usage**

ave\_control\_centrality(A)

**Arguments**

A                      An n by n matrix.

**Value**

A length n vector of average control centrality measures (Pasqualetti et al. 2014), representing the overall average control of each node in the system.

**References**

Pasqualetti F, Zampieri S, Bullo F (2014). “Controllability Metrics, Limitations and Algorithms for Complex Networks.” In *2014 American Control Conference*, 3287–3292. ISBN 978-1-4799-3274-0 978-1-4799-3272-6 978-1-4799-3271-9, doi: [10/ggkhs9](https://doi.org/10/ggkhs9).

**Examples**

```
A = matrix(c(0,-3,-2,2,-2,1,-1,2,-1), 3,3)
ave_control_centrality(A)
```

---

control\_gramian                      *Controllability Gramian*

---

**Description**

Compute the (infinite time) controllability Gramian for the discrete linear time invariant system described by  $x(t+1) = Ax(t) + Bu(t)$ . The infinite time controllability Gramian is the solution to the discrete Lyapunov equation  $AWA' - W = -BB'$ , while the finite time Gramian for time  $T$  is

$$W_t = \sum_{t=0}^T A^t B B' (A')^t$$

**Usage**

```
control_gramian(A, B, t = NA)
```

**Arguments**

A	<i>n</i> × <i>n</i> matrix.
B	<i>n</i> × <i>m</i> matrix.
t	Either NA for infinite time Gramian, or a positive non-zero integer. Defaults to NA.

**Value**

The infinite time or finite time controllability Gramian

**Examples**

```
A = matrix(c(0,-3,-2,2,-2,1,-1,2,-1), 3,3)
B = diag(3)
```

```
#Infinite time Gramian
W_inf = control_gramian(A, B)
```

```
#4 time Gramian
W_4 = control_gramian(A,B,4)
```

---

```
control_scheme_DLI_freestate
```

*Discrete Linear Time-Invariant Free Final State Classic Control Scheme*

---

**Description**

Given a system dynamics  $A$ , control input matrix  $B$ , final state weighting matrix  $S$ , intermediate state weighting matrix sequence  $Q_{seq}$ , and cost matrix sequence  $R_{seq}$ , calculates the Kalman gain sequence to minimize the LQR by time  $t_{max}$ . See section 2.2 of (Lewis et al. 2012) for details.

**Usage**

```
control_scheme_DLI_freestate(t_max, A, B, S, Q_seq, R_seq)
```

**Arguments**

t_max	Required. An integer total number of time points to determine the trajectory over
A	Required. A $p \times p$ matrix of system coefficients
B	Required. A $p \times q$ matrix of control weights

S	A $p \times p$ final state weighting matrix
Q_seq	A list of $t$ $p \times p$ intermediate state weighting matrices or a single $p \times p$ intermediate state weighting matrix
R_seq	A list of $t$ $q \times q$ intermediate cost matrices or a single $q \times q$ cost matrix

**Value**

A list containing an entry labeled gain\_seq containing either 1 or t\_max - 1 Kalman gain matrices and an entry labeled cost\_func which contains a LQR function.

**References**

Lewis FL, Vrabie DL, Syrmos VL (2012). *Optimal Control*, 3rd ed edition. Wiley, Hoboken. ISBN 978-0-470-63349-6.

**Examples**

```
A = matrix(c(0,-3,-2,2,-2,1,-1,2,-1), 3,3)

#Normalize rows to sum to 1
A = solve(diag(rowSums(A))) %% A

B = S = Q_seq = R_seq = diag(3)

CS = control_scheme_DLI_freestate(100, A, B, S, Q_seq, R_seq)
```

---

control_traj	<i>Calculate the trajectory of a discrete linear time invariant system under a given control scheme</i>
--------------	---

---

**Description**

This function is designed to work with control\_scheme objects generated by control\_scheme\_DLI\_freestate. In future versions of netcontrol this function will be used to simulate any control trajectory. For general details on control theory trajectories, see (Lewis et al. 2012).

**Usage**

```
control_traj(t_max, x_0, A, B, theta = NA, gamma = NA, control_scheme,
  delta = NA, d_nosign = F, d_toggle = F, upper_bounds = NA,
  lower_bounds = NA, u_pos = F)
```

**Arguments**

t_max	Required. An integer total number of time points to determine the trajectory over
x_0	Required. A $p$ length numeric vector of starting values
A	Required. A $pxp$ matrix of system coefficients
B	Required. A $pxq$ matrix of control weights
theta	Optional. A $pxp$ covariance matrix for state errors. If NA, state mechanics will be deterministic
gamma	Optional. A $pxp$ covariance matrix for observation errors. If NA, no observation/measurement error will be modelled.
control_scheme	Required. A list containing an entry labeled gain_seq containing either 1 or t_max - 1 Kalman gain matrices and an entry labeled cost_func which contains an appropriately constructed cost function
delta	Optional. A vector of length 2, where the first entry contains the point of saturation for control inputs, and the second entry contains the saturation value for control inputs.
d_nosign	Optional. Boolean. If TRUE and delta is not NA, control inputs are forced to be positive.
d_toggle	Optional. Boolean. If TRUE and delta is not NA, control inputs are either 0 or the saturation value.
upper_bounds	Optional. A $p$ length vector of upper bounds on state values.
lower_bounds	Optional. A $p$ length vector of lower bounds on state values.
u_pos	Optional. Boolean. If TRUE restricts control inputs to be positive,

**Details**

CAUTION: Use of saturation parameters and/or bound parameters delta, d\_nosign, d\_toggle, upper\_bound, lower\_bound, u\_pos leads to estimates of the optimal trajectory to be sub-optimal, as the Kalman gain calculations do not take any of those restrictions into account. This functionality will be added later, and this caution statement removed at that time.

**Value**

A list containing 4 entries: a 't\_max x p' state value matrix, a 't\_max x p' observation matrix, a 't\_max-1 x q' matrix of control inputs and a 't\_max' length vector of cost function values.

**References**

Lewis FL, Vrabie DL, Syrmos VL (2012). *Optimal Control*, 3rd ed edition. Wiley, Hoboken. ISBN 978-0-470-63349-6..

## Examples

```
A = matrix(c(0,-3,-2,2,-2,1,-1,2,-1), 3,3)

#Normalize rows to sum to 1
A = solve(diag(rowSums(A))) %% A

B = S = Q_seq = R_seq = diag(3)

CS = control_scheme_DLI_freestate(100, A, B, S, Q_seq, R_seq)

traj = control_traj(100, rep(100,3), A, B, control_scheme = CS)

#First 5 control inputs
print(head(traj[[3]]))
```

---

dlyap

*Discrete Lyapunov Equation Solver*

---

## Description

Computes the solution of  $AXA^T - X + W = 0$  using the Barraud 1977 approach, adapted from Datta 2004. This implementation is equivalent to the Matlab implementation of dlyap.

## Usage

```
dlyap(A, W)
```

## Arguments

A                     $n \times n$  numeric or complex matrix.  
W                     $n \times n$  numeric or complex matrix.

## Value

The solution to the above Lyapunov equation.

## References

Barraud A (1977). "A numerical algorithm to solve  $A^T X A - X = Q$ ." *IEEE Transactions on Automatic Control*, **22**(5), 883–885. ISSN 0018-9286, doi: [10/fr9gs7](https://doi.org/10.1109/9.9937), <http://ieeexplore.ieee.org/document/1101604/>.

Datta BN (2004). *Numerical methods for linear control systems: design and analysis*. Elsevier Academic Press, Amsterdam ; Boston. ISBN 978-0-12-203590-6.

**Examples**

```
A = matrix(c(0,-3,-2,2,-2,1,-1,2,-1), 3,3)
C = matrix(c(-2,-8,11,2,-6,13,-3,-5,-2), 3,3)
X = dlyap(t(A), C)

print(sum(abs(A %% X %% t(A) - X + C))
```

---

inv\_average\_control     *Trace of the Inverse Gramian*

---

**Description**

A commonly used measure of the overall controllability of a system defined by  $x(t+1) = Ax(t) + Bu(t)$ .

**Usage**

```
inv_average_control(A, B)
```

**Arguments**

A	A $n \times n$ matrix.
B	A $n \times m$ matrix.

**Value**

Trace of the inverse infinite time Gramian.

**Examples**

```
A = matrix(c(0,-3,-2,2,-2,1,-1,2,-1), 3,3)
B = diag(3)

inv_average_control(A, B)
```

---

LQR

*Linear Quadratic Regulator*

---

**Description**

Creates a function that can be used to calculate the cumulative value of the LQR for any set of states and control inputs. By setting eval to True, the LQR is immediately calculated. See (Lewis et al. 2012)

NOTE: LQR functions, as they are calculated forward in time, go to 0 by the maximum time regardless of input. This is expected behavior, but that does make using the LQR value to evaluate control efficacy somewhat difficult.



**Usage**

```
LQR(X, U, S, Q_seq, R_seq, eval = TRUE)
```

**Arguments**

X	A $txp$ matrix of state values
U	A $t - 1 \times q$ matrix of control inputs
S	A $p \times p$ final state weighting matrix
Q_seq	A list of $t$ $p \times p$ intermediate state weighting matrices or a single $p \times p$ intermediate state weighting matrix
R_seq	A list of $t$ $q \times q$ intermediate cost matrices or a single $q \times q$ cost matrix
eval	Boolean, if FALSE returns a function, if TRUE calculates the LQR series

**Value**

A function or a  $t$  length numeric vector

**References**

Lewis FL, Vrabie DL, Syrmos VL (2012). *Optimal Control*, 3rd ed edition. Wiley, Hoboken. ISBN 978-0-470-63349-6.

**Examples**

```
X = matrix(1, 100, 3)
U = matrix(-1, 99, 3)
S = Q_seq = R_seq = diag(3)

print(LQR(X,U, S, Q_seq, R_seq)[1:5])
```

---

modal\_control

*Modal Control*


---

**Description**

Calculates the modal control (Hamdan and Nayfeh 1989) of a system defined by  $x_{(t+1)} = Ax_{(t)} + Bu_{(t)}$ .

**Usage**

```
modal_control(A, B)
```

**Arguments**

A                    A  $n \times n$  matrix.  
 B                    A  $n \times m$  matrix.

**Value**

A  $m \times n$  matrix representing the control of the  $n$ th mode by the  $m$ th control input.

**Examples**

```
A = matrix(c(0,-3,-2,2,-2,1,-1,2,-1), 3,3)
B = diag(3)

modal_control(A, B)
```

---

modal\_control\_centrality  
*Modal Control Centrality*

---

**Description**

Calculates the modal control centrality of a system defined by  $x(t+1) = Ax(t)$ .

**Usage**

```
modal_control_centrality(A)
```

**Arguments**

A                    An  $n$  by  $n$  matrix.

**Value**

A length  $n$  vector of modal control centrality measures (Pasqualetti et al. 2014), representing the overall modal control of each node in the system.

**References**

Pasqualetti F, Zampieri S, Bullo F (2014). “Controllability Metrics, Limitations and Algorithms for Complex Networks.” In *2014 American Control Conference*, 3287–3292. ISBN 978-1-4799-3274-0 978-1-4799-3272-6 978-1-4799-3271-9, doi: [10/ggkhs9](https://doi.org/10/ggkhs9).

**Examples**

```
A = matrix(c(0,-3,-2,2,-2,1,-1,2,-1), 3,3)

modal_control_centrality(A)
```

---

netcontrol

*netcontrol*

---

**Description**

Description of your package

**Author(s)**

Teague Henry

# Index

ave\_control\_centrality, [3](#)  
average\_control, [2](#)  
  
control\_gramian, [3](#)  
control\_scheme\_DLI\_freestate, [4](#)  
control\_traj, [5](#)  
  
dlyap, [7](#)  
  
inv\_average\_control, [8](#)  
  
LQR, [8](#)  
  
modal\_control, [9](#)  
modal\_control\_centrality, [10](#)  
  
netcontrol, [11](#)  
netcontrol-package (netcontrol), [11](#)