# Package: LoopDetectR (via r-universe)

October 24, 2024

**Type** Package

**Title** Comprehensive Feedback Loop Detection in ODE Models

**Version** 0.1.2

**Description** Detect feedback loops (cycles, circuits) between species (nodes) in ordinary differential equation (ODE) models. Feedback loops are paths from a node to itself without visiting any other node twice, and they have important regulatory functions. Loops are reported with their order of participating nodes and their length, and whether the loop is a positive or a negative feedback loop. An upper limit of the number of feedback loops limits runtime (which scales with feedback loop count). Model parametrizations and values of the modelled variables are accounted for. Computation uses the characteristics of the Jacobian matrix as described e.g. in Thomas and Kaufman (2002) <doi:10.1016/s1631-0691(02)01452-x>. Input can be the Jacobian matrix of the ODE model or the ODE function definition; in the latter case, the Jacobian matrix is determined using 'numDeriv'. Graph-based algorithms from 'igraph' are employed for path detection.

**Imports** igraph, numDeriv

**Suggests** deSolve, knitr, markdown, remotes, rmarkdown, utils

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**Depends** R (>= 4.0.0)

**RoxygenNote** 7.1.0

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Katharina Baum [aut, cre] (<https://orcid.org/0000-0001-7256-0566>), Sandra Krüger [ctb]

**Maintainer** Katharina Baum <katharina.baum@hpi.de>

**Date/Publication** 2020-07-20 09:20:12 UTC

**Repository** https://kathbaum.r-universe.dev

**RemoteUrl** https://github.com/cran/LoopDetectR

**RemoteRef** HEAD

**RemoteSha** 6018976168428d0d68cfe1755206b64d6b51cdf3

# Contents



---

`compare_loop_list` *Compare two loop lists*

---

## Description

Compared two loop lists and returns the indices of those loops that are identical in both lists, that switch only the sign or that do not occur in both lists

## Usage

```
compare_loop_list(loop_list_a, loop_list_b)
```

## Arguments

`loop_list_a`, `loop_list_b`
: Loop lists with columns `loop` and `sign`, for example generated from `find_loops`.

## Details

Indices of loops are given with respect to the order of the loops in the first supplied loop list as well as for the second loop list. The loops are sorted to represent their loops starting from the smallest variable index (using the function `sort_loop_index`).

## Value

A list with 5 (possible empty) vectors as entries.

- `ind_a_id` - indices of the loops in the first loop list that occur identically in the second loop list
- `ind_a_switch` - indices of the loops in the first loop list that occur in the second loop list with a different sign
- `ind_a_notin` - indices of the loops in the first loop list that do not occur in the second loop list
- `ind_b_id` - indices of loops in the second loop list corresponding to the loops reported in `ind_a_id`
- `ind_b_switch` - indices of loops in the second loop list corresponding to loops reported in `ind_a_switch`.

## Examples

```
#sample Jacobian matrix of a system with 4 variables
jac_matrix <- rbind(c(-1,0,0,-1),c(1,-1,0,1),c(0,1,-1,0),c(0,0,1,-1))
#find the feedback loops of the system
loop_list <- find_loops(jac_matrix,10)
#a slightly different Jacobian matrix of the system with 4 variables
jac_matrix_alt <- rbind(c(-1,0,0,1),c(1,-1,0,-1),c(0,1,-1,0),c(0,0,1,-1))
#find the feedback loops of the system
loop_list_alt <- find_loops(jac_matrix_alt,10)
#compare the loop lists
comp_loop_list <- compare_loop_list(loop_list,loop_list_alt)
#loops that switch sign
comp_loop_list[['ind_a_switch']]
```

---

`find_edge` *Detecting loops with a certain edge*

---

## Description

Finds those loops in a loop list that contain a regulation from a certain variable (source node) to a certain variable (target node).

## Usage

```
find_edge(loop_list, source_node, target_node)
```

### Arguments

| | |
|---|---|
| loop_list | Dataframe with a column loop that contains the lists of loops, e.g. obtained from find_loops. |
| source_node | Index of the variable that is the source of the queried interaction, i.e. that regulates the target node. |
| target_node | Index of the variable that is the target of the queried interaction, i.e. that is regulated by the source node. |

### Value

A vector that gives the indices in the loop list of those loops that contain the indicated edge.

### Examples

```
#sample Jacobian matrix of a system with 4 variables
jac_matrix <- rbind(c(-1,0,0,-1),c(1,-1,0,1),c(0,1,-1,0),c(0,0,1,-1))
#find the feedback loops of the system
loop_list <- find_loops(jac_matrix,10)
#find the loops containing the regulation from variable 3 to variable 4
inds_3_to_4 <- find_edge(loop_list,3,4)
```

---

## find_loops *Loop detection in a matrix*

---

### Description

Given the Jacobian matrix of an ODE system or the adjacency matrix of a graph, this function determines all loops in the system up to the maximal number supplied.

### Usage

```
find_loops(jacobian, max_num_loops = 1e+05)
```

### Arguments

| | |
|---|---|
| jacobian | Square Jacobian matrix of an ODE system or the adjacency matrix of a graph; captures interactions such that entry (i,j) is negative (positive) if variable j regulates variable i negatively (positively). |
| max_num_loops | Positive numeric value indicating the maximal number of loops that are reported. Default: $10^5$. |

## Details

The input matrix delivers the directed interactions in the ODE system; if entry `(i,j)` is non-zero it means that variable (or node) `i` is regulated by variable (node) `j`. Johnson's algorithm for path detection as well as Tarjan's algorithm for detecting strongly connected components are used as implemented in the igraph package (functions: `all_simple_paths`, `components`) . If the maximal number of loops, max_num_loops, is reached, no warning is issued. It is very probable that not all feedback loops of the system have been found. Running the function multiple times with re-ordered jacobian as input can enable detection of alternative feedback loops while limiting the runtime and output size of single runs. If columns of the Jacobian are named, the identification is given by the attribute `node_ids`, `attr(result,"node_ids")`.

## Value

A data.frame with three columns: `loop`, `length`, `sign` containing up to `max_num_loops` loops of the systems defined by matrix `jacobian`. Each entry in the loop column is a list of identifiers that correspond to the indices of the variable in the Jacobian matrix and denote in which order the variables form the loop.

## See Also

`find_loops_noscc`, `find_loops_vset`

## Examples

```
#sample Jacobian matrix of a system with 4 variables
jac_matrix <- rbind(c(-1,0,0,-1),c(1,-1,0,1),c(0,1,-1,0),c(0,0,1,-1))
#find the first 5 feedback loops of the system
loop_list <- find_loops(jac_matrix,5)
```

---

# find_loops_noscc *Loop detection in a matrix*

---

## Description

Given the Jacobian matrix of an ODE system or the adjacency matrix of a graph, this function determines all loops in the system up to the maximal number supplied. No decomposition into strongly connected components is performed.

## Usage

```
find_loops_noscc(jacobian, max_num_loops = 1e+05)
```

## Arguments

| | |
|---|---|
| `jacobian` | Square Jacobian matrix of an ODE system or the adjacency matrix of a graph; captures interactions such that entry `(i,j)` is negative (positive) if variable `j` regulates variable `i` negatively (positively). |
| `max_num_loops` | Positive numeric value indicating the maximal number of loops that are reported. Default: $10^5$. |

## Details

The input matrix delivers the directed interactions in the ODE system; if entry `(i,j)` is non-zero it means that variable (or node) `i` is regulated by variable (node) `j`. Johnson's algorithm for path detections from the igraph package (function: `all_simple_paths`) is used. No decomposition into strongly connected components is employed which could be beneficial for smaller systems (compared to `find_loops`). The queried graph is increased stepwise leading to the output of loops in a certain order determined by the order of occurrence in the Jacobian matrix:

- first the self-loops,
- then feedback loops incorporating only the first and second species of the Jacobian,
- then feedback loops incorporating the third and at most also the first and second species of the jacobian, etc.

If the maximal number of loops, max_num_loops, is reached, no warning is issued. It is very probable that not all feedback loops of the system have been found. Up to which species this function searched before stopping due to reaching the maximal allowed loop number can be inferred from the last exported feedback loop. Running the function multiple times with re-ordered jacobian as input can enable detection of alternative feedback loops while limiting the runtime and output size of single runs. If columns of the Jacobian are named, the identification is given by the attribute `node_ids`, `attr(result,"node_ids")`.

## Value

A data.frame with three columns: `loop`, `length`, `sign` containing up to `max_num_loops` loops of the systems defined by matrix `jacobian`. Each entry in the loop column is a list of identifiers that correspond to the indices of the variable in the Jacobian matrix and denote in which order the variables form the loop.

## See Also

`find_loops`, `find_loops_vset`

## Examples

```
#sample Jacobian matrix of a system with 4 variables
jac_matrix <- rbind(c(-1,0,0,-1),c(1,-1,0,1),c(0,1,-1,0),c(0,0,1,-1))
#find the first 5 feedback loops of the system
loop_list <- find_loops_noscc(jac_matrix,5)
```

find_loops_vset *Loop detection for an ODE model at multiple sets of variables*

## Description

Determines loop lists for an ODE system given by a function and at multiple sets of variables. Loop lists are reported if signs of Jacobian matrix have changed.

## Usage

```
find_loops_vset(
  fun,
  vset,
  ...,
  max_num_loops = 1e+05,
  compute_full_list = TRUE
)
```

## Arguments

| | |
|---|---|
| fun | Function defining the ODE system, returns the vector $dx/dt$. May depend on further parameters in .... |
| vset | List of variable values at which the loops are determined. |
| ... | Further parameters except variable values to the function fun, none called x. |
| max_num_loops | Positive numeric value indicating the maximal number of loops that are reported in a loop list. Default: $10^5$. |
| compute_full_list | Logical value indicating whether for each Jacobian matrix with any different sign the loop list is computed (TRUE, default), or whether further checks are performed to ensure that loops may be altered. |

## Details

The supplied function can take more arguments, but only the variables are allowed to be named x (they can also be named differently). The Jacobian matrices are computed for each of the variable values defined in vset using the jacobian function from the NumDeriv package with option method = 'complex', i.e. using a complex-step approach. If compute_full_list = TRUE (default), loop lists are not re-computed for Jacobians that clearly do not allow for altered loop lists. This is the case if no new regulation appear and only signs of regulations are altered that are not member of any loop. Loop lists can still be identical for different Jacobians, e.g. if two sign switches occur that are both affecting the same loops.

If there is only one class of Jacobian matrix (i.e. the signs of the Jacobian matrix are the same for all entries in vset), loop_rep and jac_rep will have only one entry each. The number of entries for loop_rep_index and jac_rep_index corresponds to the length of vset. Only if compute_full_list is set to FALSE, loop_rep can contain fewer elements than jac_rep, otherwise both have the same number of elements.

## Value

A list with four entries:

- loop_rep List of loop lists.
- loop_rep_index Vector of integer numbers returning the index of the loop list in loop_rep belonging to each entry in vset.
- jac_rep List of signed Jacobian matrices.
- jac_rep_index Vector of integer numbers returning the index of the Jacobian matrix in jac_rep belonging to each entry in vset.

## Examples

```
#default call to determine loops from an ODE model given by a function
#read in example functions
data("func_POSm4")
#the loaded function func_POSm4 takes arguments t, x, klin, knonlin
res_tab <- find_loops_vset(func_POSm4,vset=list(c(1,1,1,1)),t=1,
klin=c(1,2,0.5,1,2,0.1,3,2,3),knonlin=c(1,2))
#computed loop list:
res_tab$loop_rep[[1]] #or res_tab[[1]][[1]]

#determine loops from an ODE model over the course of a solution
#read in the example function defining the bacterial cell cycle
data("func_li08")
#kinetic parameter values are defined within the function
#read in a set of variable values (the solution of func_li08 with events)
data("li08_solution")
#transform the solution (columns: variables) to the correct list format
#and remove the time (first column)
li08_sol_list <- as.list(as.data.frame(t(li08_solution[,-1])))
res_tab <- find_loops_vset(func_li08,vset=li08_sol_list,t=1,
compute_full_list=FALSE)
```

---

# func_li08 *Example ODE function: bacterial cell cycle.*

---

## Description

The file contains the function definition an ordinary differential equation model of Caulobacter crescentus cell cycle as proposed by Li et al., 2008. It has 18 variables.

## Usage

```
func_li08
```

### Format

R file with definition of function func_li08 that takes as input arguments time t (dimension 1), and variable values y (dimension 18). the kinetic parameters are defined within the function.

### Details

The Caulobacter cell cycle model function will only give the solution as shown in the publication [Li et al., 2008] if the change in variables at defined events are taken into account. Please refer to the original reference for details.

### Source

The Caulobacter cell cycle model was proposed in Li S, Brazhnik P, Sobral B, Tyson JJ. A Quantitative Study of the Division Cycle of Caulobacter crescentus Stalked Cells. Plos Comput Biol. 2008;4(1):e9. The function corresponds to the MATLAB function modelwtin(t,y) as given on http://mpf.biol.vt.edu/research/caulobacter/SWST/pp/.

---

## func_POSm4 *Example ODE function: chain model with positive regulation.*

---

### Description

The file contains the function definition an ordinary differential equation model of a chain model of 4 variables with positive feedback.

### Usage

```
func_POSm4
```

### Format

R file with definition of function func_POSm4 that takes as input arguments time t (dimension 1), variable values x (dimension 4), and kinetic parameter values klin (dimension 8) and knonlin (dimension 2).

### Source

The chain model was used in Baum K, Politi AZ, Kofahl B, Steuer R, Wolf J. Feedback, Mass Conservation and Reaction Kinetics Impact the Robustness of Cellular Oscillations. PLoS Comput Biol. 2016;12(12):e1005298.

## li08_solution *Solution for the cell cycle model related to func_li08*

### Description

The file contains the solution over time (3 oscillatory cycles) for the ordinary differential equation model as given in func_li08. In addition, events as constructed in the original publication [Li et al., 2008] are considered.

### Usage

```
li08_solution
```

### Format

A dataframe with 634 rows and 19 columns

**time** time variable

**y1** first variable value

**y2** second variable value,

**etc.** etc.

**y18** 18th variable value

### Source

The Caulobacter cell cycle model was proposed in Li S, Brazhnik P, Sobral B, Tyson JJ. A Quantitative Study of the Division Cycle of Caulobacter crescentus Stalked Cells. Plos Comput Biol. 2008;4(1):e9. The solutions were generated with MATLAB using the functions accompanying the above reference on http://mpf.biol.vt.edu/research/caulobacter/SWST/pp/.

## loop_summary *Summary of a loop list*

### Description

Summarizes the loops in a loop list by their length and sign, returns an overview table of the numbers of all, negative and positive loops divided by their lengths.

### Usage

```
loop_summary(loop_list, column_val = "length")
```

### Arguments

| | |
|---|---|
| loop_list | List of loops as dataframe with columns length, sign. |
| column_val | String indicating the orientation of the summary table. By default, rows of the results table are the sign of the loops, columns are loop lengths. If column_val is set to "sign", columns and rows are exchanged. |

### Details

Lengths are abbreviated by len_1, len_2, len_3 etc., signs are abbreviated by pos for positive, neg for negative loops. The table contains entries for each loop length from 1 to the maximal loop length encountered in the table, and zeros are filled in if no loops of a certain length exist in the table.

### Examples

```
#sample Jacobian matrix of a system with 4 variables
jac_matrix <- rbind(c(-1,0,0,-1),c(1,-1,0,1),c(0,1,-1,0),c(0,0,1,-1))
#find the feedback loops of the system
loop_list <- find_loops(jac_matrix,10)
#loop summary table
loop_sum_tab <- loop_summary(loop_list)
```

---

## sort_loop_index *Sort loop indices*

---

### Description

Changes the loop representation such that every loop starts with the smallest node index. Returns a loop list of the same dimensions, only column loop will be altered.

### Usage

```
sort_loop_index(loop_list)
```

### Arguments

| | |
|---|---|
| loop_list | Dataframe with a column loop that contains the lists of loops, e.g. obtained from find_loops(). |

### See Also

compare_loop_list

## Examples

```
#sample Jacobian matrix of a system with 4 variables
jac_matrix <- rbind(c(-1,0,0,-1),c(1,-1,0,1),c(0,1,-1,0),c(0,0,1,-1))
#find the feedback loops of the system
loop_list <- find_loops(jac_matrix,10)
#sort the loop indices to start with the smallest
sorted_loop_list <- sort_loop_index(loop_list)
```

# Index