Package 'linkR'

October 1, 2016

Date 2016-10-01

Type Package

Title 3D Lever and Linkage Mechanism Modeling

Description Creates kinematic and static force models of 3D levers and linkage mechanisms, with particular application to the fields of engineering and biomechanics.

Version 1.1.1

Depends R (>= 2.11.0), svgViewR (>= 1.2)

Author Aaron Olsen

Maintainer Aaron Olsen <aarolsen@gmail.com>

Repository CRAN

URL https://aaronolsen.github.io/software/linkr.html

License GPL (>= 2)

NeedsCompilation no

Date/Publication 2016-10-01 17:53:52

R topics documented:

linkR-package *3D lever and linkage mechanism modeling*

Description

The linkR package for R provides a toolkit for creating models of 2D and 3D linkage networks, predicting motion, force and torque outputs and calculating linkage mechanical properties such as motion, force and torque transmission ratios. Users in the fields of engineering and biomechanics will find linkR particularly useful. linkR works with the R package svgViewR, enabling users to create 3D interactive animations of linkage models. Please see linkR Resources for a tutorial on how to use the linkR package and the linkR example gallery for several examples (code and animations) of different linkages that can be created using linkR. For instructions on how to view svgViewR animations, please see svgViewR interactive commands.

Details

Package:	linkR
Type:	Package
Version:	1.1.1
Date:	2016-10-01
License:	GPL (>= 2)

Author(s)

Aaron Olsen Maintainer: Aaron Olsen <aarolsen@gmail.com>

animateLinkage *Animates a lever or linkage*

Description

This function animates a lever or linkage mechanism according to specified translational or rotational motion of the input link.

Usage

```
animateLinkage(linkage, input.param, input.joint=NULL,
               check.inter.joint.dist = TRUE, check.joint.cons = TRUE,
               check.inter.point.dist = TRUE, print.progress = FALSE)
```

Arguments

`linkage`	A linkage object. This is a list of class `"linkage"` created by `defineLinkage`.
`input.param`	A list containing the input parameters for the linkage. The number of list elements corresponds to the number of joints for which input motion is specified. Thus, a one degree-of-freedom mechanism should be of length one; a two degree-of-freedom mechanism of length two, etc.
`input.joint`	A vector specifying the joint(s) at which the input motion is to be applied. Currently, input motion can only be specified at grounded joints.
`check.inter.joint.dist`	A logical indicating whether a check should be run to verify that the distances among connected joints is constant. A warning message is displayed if the distances are non-constant.
`check.joint.cons`	A logical indicating whether a check should be run to verify that the joint constraints are maintained through the simulation. This feature is currently in development and does not work for all joint types.
`check.inter.point.dist`	A logical indicating whether a check should be run to verify that the distances among points on the same body are constant. A warning message is displayed if the distances are non-constant.
`print.progress`	A logical indicating whether to print progress while running.

Details

Please see linkR Resources for a tutorial on how to use the linkR package and the linkR example gallery for several examples (code and animations) of different linkages that can be created using linkR.

Value

a list of class `"linkage"` having the same elements as the input `linkage` with some exceptions, including:

`joint.coor`	The joint matrix will be converted to an array, where `dim(joints)[3]` is equal to the number of animation iterations.
`points`	If provided in the input, the point matrix will be converted to an array, where `dim(points)[3]` is equal to the number of animation iterations.

Author(s)

Aaron Olsen

See Also

`defineLinkage`

cprod *Computes the cross product of two vectors*

Description

Returns the cross product of two vectors using either the right-hand or left-hand convention.

Usage

```
cprod(u, v, h = "right")
```

Arguments

u	a vector of length 3
v	a vector of length 3
h	whether the right-hand, "right", or left-hand, "left", convention is to be used. The right-hand convention is default.

Details

The cross product vector is a vector orthogonal to the two input vectors.

Value

the cross product vector (also of length 3)

Author(s)

Aaron Olsen

Examples

```
## DEFINE TWO 3D VECTORS
u <- c(1, 0, 0)
v <- c(0, 1, 0)

## FIND THE CROSS PRODUCT
cprod(u, v)
```

`defineLinkage` *Defines a lever or linkage*

Description

This function takes the joint positions, types and constraints of a lever or linkage mechanism or a set of minimum parameters and creates a list of class `"linkage"` that forms the basic object for model creation and analysis.

Usage

```
defineLinkage(joint.coor, joint.types, joint.cons,
              joint.conn = NULL, link.points = NULL, link.assoc = NULL,
              link.names = NULL, ground.link = NULL, path.connect = NULL,
              lar.cons = NULL)
```

Arguments

`joint.coor`	A matrix of 2D or 3D coordinates that are the joints of a lever or linkage mechanism.
`joint.types`	A vector of single letters indicating the type of constraints acting on each joint. Supported types are "R", "U", "L" and "P". See Details.
`joint.cons`	A list or matrix of 3D constraint vectors that correspond to each joint.
`joint.conn`	A two-column matrix having the same number of rows as `joint.coor`, specifying the two links that each joint connects.
`link.points`	A matrix of points associated with one or more of the links in the linkage (optional).
`link.assoc`	A vector of integers or rownames indicating the link with which each row in `points` is associated. Required if `points` is non-`NULL` but otherwise optional.
`link.names`	A vector of the link names (optional).
`ground.link`	A numeric or link name specifying the ground link of the linkage.
`path.connect`	A list of vectors, each specifying the points to connect (in sequence) with lines when drawing. This parameter is only used when calling the function `drawLinkage`.
`lar.cons`	A list specifying long-axis rotation constraints on links with S-joints on each end. This feature will eventually be added into the `joint.conn` input parameter as network connections.

Details

Please see linkR Resources for a tutorial on how to use the linkR package and the linkR example gallery for several examples (code and animations) of different linkages that can be created using linkR.

Value

a list of class "linkage" with the following elements:

joint.coor	A matrix of 2D or 3D coordinates that are the joints of a lever or linkage mechanism.
joint.cons	A vector of single letters indicating the type of constraints acting on each joint.
joint.types	A list or matrix of 3D constraint vectors that correspond to each joint.
joint.links	A matrix specifying the joints connecting to each link. Used by the animateLinkage function.
joint.paths	"Path fragments" along the linkage network, used by the animateLinkage function to solve for the position of unresolved joints.
joint.conn	A two-column matrix having the same number of rows as joint.coor, specifying the two links that each joint connects.
joint.init	The initial position of the joints in the linkage.
ground.joints	The ground joints in the linkage.
points	A matrix of points associated with any of the links in the linkage.
points.assoc	The links with which each point is associated.
link.assoc	A vector of integers or rownames indicating the link with which each row, if input.
link.names	A vector of the link names.
point.connect	A list of vectors, each of which specifies linkage-associated points to connect in sequence with a path. This is only used by drawLinkage when creating the linkage visualization.
link.lcs	Local coordinate systems assigned to each link which will be transformed with the associated link for kinematic analysis.
lar.cons	Constraints on long-axis rotation.
num.links	The number of links in the linkage.
dof	The number of degrees of freedom of the linkage. Currently, this may not be accurately estimated for all linkages.

Author(s)

Aaron Olsen

See Also

animateLinkage, drawLinkage

drawLinkage *Draws a lever or linkage*

Description

This function creates a visualization of a linkage, with options to create a static plot or an interactive, 3D visualization.

Usage

```
drawLinkage(linkage, method="svgViewR", file=NULL, animate = TRUE,
    animate.duration = 1, animate.reverse = FALSE, animate.repeat = -1,
    path.connect=NULL, connect.joints=TRUE, window.title='Linkage Viewer',
    joint.col.fill="white", joint.col.stroke="black", joint.cex=1.5,
    joint.lwd=2, point.col.fill="black", point.col.stroke="black",
    point.cex=1, point.lwd=2, path.col.fill=NA, path.opacity.fill=1,
    path.opacity.stroke=1, path.col.stroke="black", path.lwd = 1,
    add = FALSE, ...)
```

Arguments

linkage	A linkage object. This can be the output of defineLinkage (to visualize the initial static conformation of the linkage) or animateLinkage (to visualize the animated linkage).
method	The method to use in creating the visualization. The default method uses the package svgViewR to create an interactive, 3D visualization as an '.html' file that can be opened in any major web browser. The alternative method is "plot", which draws the linkage in an R graphics window (as a static visualization). Note that the "plot" method is still in development and may have limited functionality. The use of the "svgViewR" method (default) is recommended.
file	A filename for the '.html' file if method is "svgViewR". For method "plot", a filename with the extension of the desired save-as filetype. Supported filetypes for method "plot" are: .bmp, .png, .jpg (or .jpeg), .tiff, .eps.
animate	A logical indicating whether the linkage should be drawn as an animation or as a static visualization with all frames superimposed. This only applies if the input parameter linkage is an object created by animateLinkage. An input parameter linkage created by defineLinkage will only have a single frame (the initial linkage conformation).
animate.duration	The number of seconds during which the entire linkage animation will playback in the viewer. Note that for linkages with a large number of associated points processing speed may reduce the speed with which the animation can play such that the duration will actually be greater than that specified here.
animate.reverse	Whether the animation should play both forward and reverse.

`animate.repeat` The number of times the animation should repeat. A value of -1 (default) causes the animation to play non-stop.

`path.connect` A list of vectors, each of which specifies linkage-associated points to connect in sequence with a path. If this is specified in `defineLinkage` then it does not have to be specified here as it will be already contained within the `linkage` object.

`connect.joints` A logical indicating whether the joints should be connected in the visualization. The connections will correspond to those specified by the input parameter `joint.conn` to `defineLinkage`.

`window.title` For `method` "svgViewR", the title that will appear at the top of the '.html' visualization file.

`joint.col.fill` The fill color of the points representing linkage joints.

`joint.col.stroke` The stroke (outline) color of the points representing linkage joints.

`joint.cex` The size of the points representing linkage joints.

`joint.lwd` The thickness of the stroke (outline) of the points representing linkage joints.

`point.col.fill` The fill color of the linkage-associated points.

`point.col.stroke` The stroke (outline) color of the linkage-associated points.

`point.cex` The size of the linkage-associated points.

`point.lwd` The thickness of the stroke (outline) of the linkage-associated points.

`path.col.fill` The fill color of the paths specified by `path.connect`. By default, the paths are not filled.

`path.opacity.fill` The fill opacity of the paths specified by `path.connect`.

`path.opacity.stroke` The stroke (outline) opacity of the paths specified by `path.connect`.

`path.col.stroke` The color of the stroke (outline) opacity of the paths specified by `path.connect`.

`path.lwd` The thickness of the stroke (outline) opacity of the paths specified by `path.connect`.

`add` Logical indicating whether linkage should be added to an existing visualization. This option is still in developement.

`...` Additional parameters to be passed to plot device opening functions (e.g. 'jpeg', 'png', 'eps') in the case of `method` "plot".

Details

Please see linkR Resources for a tutorial on how to use the linkR package and the linkR example gallery for several examples (code and animations) of different linkages that can be created using linkR.

Value

NULL

Author(s)

Aaron Olsen

See Also

animateLinkage, defineLinkage

`rotationMatrixZYX` *Returns a matrix to rotate points along the z-, y- and x-axes*

Description

This function returns a rotation matrix that rotates a three-column point matrix about the z-, y- and x-axes, in that order. The three angles of rotation can be specified by a single vector of length three or three separate parameters.

Usage

```
rotationMatrixZYX(t, t2 = NULL, t3 = NULL)
```

Arguments

`t`	angle (in radians) to rotate around the z-axis or a vector of three angles to rotate around the z-, y- and x-axes, in that order.
`t2`	if `t` is a single numeric, this is the angle (in radians) to rotate around the y-axis.
`t3`	if `t` is a single numeric, this is the angle (in radians) to rotate around the x-axis.

Value

a 3x3 rotation matrix.

Author(s)

Aaron Olsen

References

http://en.wikipedia.org/wiki/Rotation_matrix#In_three_dimensions

Examples

```
## SPECIFY 3D POINT SET
m <- matrix(c(0,0,0, 1,2,1, 3,0,3, -2,4,1), nrow=4, ncol=3, byrow=TRUE)

## ROTATE 180 DEGREES ABOUT THE Z AXIS
## X AND Y VALUES ARE OPPOSITE AND Z VALUES UNCHANGED
m %*% rotationMatrixZYX(pi, 0, 0)

## ROTATE 180 DEGREES ABOUT THE X AXIS
## Y AND Z VALUES ARE OPPOSITE AND X VALUES UNCHANGED
m %*% rotationMatrixZYX(0, 0, pi)

## ROTATE 90 DEGREES ABOUT THE Z-, THEN Y-, THEN X-AXIS
m %*% rotationMatrixZYX(pi/2, pi/2, pi/2)

## ROTATE 90 DEGREES ABOUT THE Z-, THEN Y-, THEN X-AXIS
m %*% rotationMatrixZYX(c(pi/2, pi/2, pi/2))
```

Index