

Open Loop Dynamic Library

Generated by Doxygen 1.7.1

Wed Jul 17 2024 10:21:19

Contents

1	Class Index	1
1.1	Class Hierarchy	1
2	Class Index	3
2.1	Class List	3
3	File Index	5
3.1	File List	5
4	Class Documentation	7
4.1	Action Class Reference	7
4.1.1	Constructor & Destructor Documentation	8
4.1.1.1	Action	8
4.1.2	Member Function Documentation	8
4.1.2.1	Get_Info	8
4.1.3	Member Data Documentation	8
4.1.3.1	Action_Base	8
4.1.3.2	Action_Name	8
4.1.3.3	Action_Point	8
4.1.3.4	Action_Vector	8
4.2	Base Class Reference	8
4.2.1	Detailed Description	9
4.2.2	Constructor & Destructor Documentation	10
4.2.2.1	Base	10
4.2.3	Member Function Documentation	10
4.2.3.1	Get_Info	10
4.2.3.2	Rotation_Change_Base_Matrix	10
4.2.4	Member Data Documentation	11
4.2.4.1	Axis_Labels	11

4.2.4.2	Base_Name	11
4.2.4.3	M_B_E	11
4.3	handle Class Reference	11
4.4	Point Class Reference	11
4.4.1	Constructor & Destructor Documentation	12
4.4.1.1	Point	12
4.4.2	Member Function Documentation	12
4.4.2.1	Get_Info	12
4.4.3	Member Data Documentation	12
4.4.3.1	Point_Coordinates_From_Canonical	12
4.4.3.2	Point_Name	12
4.5	Rigid_Body Class Reference	12
4.5.1	Detailed Description	13
4.5.2	Constructor & Destructor Documentation	14
4.5.2.1	Rigid_Body	14
4.5.3	Member Function Documentation	14
4.5.3.1	Get_Info	14
4.5.4	Member Data Documentation	15
4.5.4.1	G_Point_From_Canonical	15
4.5.4.2	Inertial_Tensor	15
4.5.4.3	Mass	15
4.5.4.4	Rigid_Body_Base	15
4.5.4.5	Rigid_Body_Name	15
4.6	System Class Reference	15
4.6.1	Detailed Description	20
4.6.2	Constructor & Destructor Documentation	20
4.6.2.1	System	20
4.6.3	Member Function Documentation	20
4.6.3.1	Angular_Velocity	20
4.6.3.2	Change_Basis_Matrix	21
4.6.3.3	Create_New_Action	21
4.6.3.4	Create_New_Base	21
4.6.3.5	Create_New_Coordinate_System	22
4.6.3.6	Create_New_Generalized_Coordinate	22
4.6.3.7	Create_New_Point	22
4.6.3.8	Create_New_Rigid_Body	23

4.6.3.9	dot_A_dot_q	23
4.6.3.10	Get_A_Matrix_Handle	23
4.6.3.11	Get_A_tk_xk	23
4.6.3.12	Get_Action_in_G	24
4.6.3.13	Get_Action_Info	24
4.6.3.14	Get_Base_Info	24
4.6.3.15	Get_Coordinate_System_Info	24
4.6.3.16	Get_d_A_tk_xk	24
4.6.3.17	Get_d_Kinetic_Energy_Quadratic_Matrix	24
4.6.3.18	Get_Generalized_Action	24
4.6.3.19	Get_Generalized_Coordinates_Info	24
4.6.3.20	Get_input_tk_xk	24
4.6.3.21	Get_Kinetic_Energy_Quadratic_Matrix	25
4.6.3.22	Get_Point_Info	25
4.6.3.23	Get_Rigid_Body_Info	25
4.6.3.24	Get_System_A_Matrix_Handles	25
4.6.3.25	Get_System_d_Kinetic_Energy_Quadratic_Matrix	25
4.6.3.26	Get_System_Generalized_Actions	25
4.6.3.27	Get_System_Kinetic_Energy_Quadratic_Matrix	25
4.6.3.28	Get_System_Transformation_Matrix_Handle	26
4.6.3.29	Get_System_Virtual_Velocity_q_uv_handle	26
4.6.3.30	Get_tau_tk_xk	26
4.6.3.31	Get_Transformation_Matrix_Handle	26
4.6.3.32	Get_Two_Points_Vector	26
4.6.3.33	Get_Two_Points_Vector_Base	26
4.6.3.34	Get_Virtual_Velocity_q_Component	27
4.6.3.35	Get_Virtual_Velocity_q_uv_handle	27
4.6.3.36	Lagrangian_Rigth_Hand_Side	27
4.6.3.37	order_reduction	27
4.6.3.38	System_Create_Action_Table	27
4.6.3.39	System_Create_Bases_Table	28
4.6.3.40	System_Create_Canonical_action	28
4.6.3.41	System_Create_Canonical_Base	28
4.6.3.42	System_Create_Canonical_Coordinate_System	28
4.6.3.43	System_Create_Canonical_Point	28
4.6.3.44	System_Create_Canonical_Rigid_Body	28

4.6.3.45	System_Create_Coordinate_System_Table	28
4.6.3.46	System_Create_Generalized_Coordinates_Table	28
4.6.3.47	System_Create_Points_Table	29
4.6.3.48	System_Create_Rigid_Body_Table	29
4.6.3.49	unify_symbolic_system	29
4.6.3.50	unify_system	29
4.6.4	Member Data Documentation	29
4.6.4.1	A	29
4.6.4.2	A_Matrix_Fun_Handles	29
4.6.4.3	A_uv	30
4.6.4.4	b	30
4.6.4.5	b_uv	30
4.6.4.6	d_A	30
4.6.4.7	d_A_uv	30
4.6.4.8	q	30
4.6.4.9	q_variables	30
4.6.4.10	System_Actions	30
4.6.4.11	System_Bases	30
4.6.4.12	System_Canonical_Action	30
4.6.4.13	System_Canonical_Base	30
4.6.4.14	System_Canonical_Coordinate_System	30
4.6.4.15	System_Canonical_Generalized_Coordinate	30
4.6.4.16	System_Canonical_Input	30
4.6.4.17	System_Canonical_Point	30
4.6.4.18	System_Canonical_Rigid_Body	30
4.6.4.19	System_Coordinate_Systems	30
4.6.4.20	System_Generalized_Coordinates	30
4.6.4.21	System_Inputs	30
4.6.4.22	System_Name	30
4.6.4.23	System_Points	31
4.6.4.24	System_Rigid_Bodies	31
4.6.4.25	Transformation_Matrix_Handles	31
4.6.4.26	u	31
4.6.4.27	uv	31
4.6.4.28	v	31
4.6.4.29	Virtual_Velocity_Components_fun_handles	31

4.6.4.30	vu	31
5	File Documentation	33
5.1	C:/DoxygenMatlab/+Dynamic_Library/+Classes/@Action/Action.m File Reference	33
5.2	C:/DoxygenMatlab/+Dynamic_Library/+Classes/@Base/Base.m File Reference	33
5.2.1	Detailed Description	33
5.3	C:/DoxygenMatlab/+Dynamic_Library/+Classes/@Point/Point.m File Reference	33
5.4	C:/DoxygenMatlab/+Dynamic_Library/+Classes/@Rigid_Body/Rigid_Body.m File Ref- erence	34
5.4.1	Detailed Description	34
5.5	C:/DoxygenMatlab/+Dynamic_Library/System.m File Reference	34
5.5.1	Detailed Description	34

Chapter 1

Class Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

handle	11
Action	7
Base	8
Rigid_Body	12
System	15
Point	11

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Action	7
Base (Base Class for the creation of orthonormal vectorial bases)	8
handle	11
Point	11
Rigid_Body (Rigid Body Class for the creation of Rigid Body objects)	12
System (System Class for the creation of System objects)	15

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

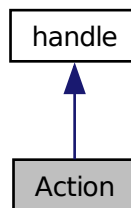
C:/DoxygenMatlab/+Dynamic_Library/System.m (System Class description)	34
C:/DoxygenMatlab/+Dynamic_Library/+Classes/@ Action/Action.m	33
C:/DoxygenMatlab/+Dynamic_Library/+Classes/@Base/Base.m (Base Class description) . . .	33
C:/DoxygenMatlab/+Dynamic_Library/+Classes/@Point/Point.m	33
C:/DoxygenMatlab/+Dynamic_Library/+Classes/@Rigid_Body/Rigid_Body.m (Rigid Body Class description)	34

Chapter 4

Class Documentation

4.1 Action Class Reference

Inheritance diagram for Action:



Public Member Functions

- function [Action](#) (in Name, in varargin)
Action Class constructor.
- function [Get_Info](#) (in obj, in Info)

Public Attributes

- Property [Action_Name](#)
- Property [Action_Point](#)
- Property [Action_Base](#)
- Property [Action_Vector](#)

4.1.1 Constructor & Destructor Documentation

4.1.1.1 function Action (in *Name*, in *varargin*)

[Action](#) Class constructor.

Parameters

obj,: [System](#) in which to introduce a new action

Name,: Name of the new action

Point,: [Point](#) (object) of application of the action.

Base,: [Base](#) (object) in which the action components are given.

Vector,: Components of the [Action](#). The action components can be symbolic or double and they are given by 3 torque components and 3 force components. Eg: [Mx;My;Mz;Fx;Fy;Fz].

4.1.2 Member Function Documentation

4.1.2.1 function Get_Info (in *obj*, in *Info*)

4.1.3 Member Data Documentation

4.1.3.1 Property Action_Base

4.1.3.2 Property Action_Name

4.1.3.3 Property Action_Point

4.1.3.4 Property Action_Vector

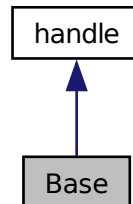
The documentation for this class was generated from the following file:

- C:/DoxygenMatlab/+Dynamic_Library/+Classes/@Action/[Action.m](#)

4.2 Base Class Reference

[Base](#) Class for the creation of orthonormal vectorial bases.

Inheritance diagram for Base:



Public Member Functions

- function [Base](#) (in Name, in varargin)
Base Class constructor.
- function [Get_Info](#) (in obj, in Info)
Get_Info is a public function to get the properties of the [Base](#) objects.

Static Public Member Functions

- static function [Rotation_Change_Base_Matrix](#) (in axis, in angle)
Rotation_Change_Base_Matrix is a Static function that calculates a matrix to change the components of a vector in a moving base (BM) to a fixed base (BF).

Public Attributes

- Property [Base_Name](#)
String that indicates the name of the base object.
- Property [Axis_Labels](#)
[3x1] symbolic array that represents the labels for the vectors of the base.
- Property [M_B_E](#)
[3x3] symbolic or double array that represents the matrix to change the coordinates of a vector in the canonical base (E) to the actual base (B).

4.2.1 Detailed Description

[Base](#) Class for the creation of orthonormal vectorial bases.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 function Base (in *Name*, in *varargin*)

[Base](#) Class constructor.

The [Base](#) constructor can initialize the object in one of two ways: either as the canonical basis $[1, 0, 0]$, $[0, 1, 0]$, and $[0, 0, 1]$, or through rotation with respect to another base. This means that the base can be directly set to the standard orthonormal basis vectors, which are the default coordinate axes, or it can be created by rotating an already existing base.

Parameters

Name

axis_labels(optional) $[3 \times 1]$ symbolic array that contains the labels of the 3 vectors that represents the base. If this parameter is not specified the default *axis_labels* will be $[x, y, z]^T$.

father_base(optional) The parent [Base](#) object defines the base from which the constructor will create a new [Base](#) object. From this parent base, a new base is generated through a simple rotation. The simple rotation can only be performed in the direction indicated by the right-hand rule for each of the vectors of the parent base. If *father_base* parameter is not defined, the default parent base is $[1, 0, 0]$, $[0, 1, 0]$, and $[0, 0, 1]$.

axis_rotation(optional) The simple rotation can only be performed in the direction indicated by the right-hand rule for each of the vectors of the parent base. The *axis_rotation* parameter defines the rotation axis with a string "1", "2" or "3" defining the parent basis vector to use. If *axis_rotation* parameter is not defined, the default axis of rotation "1".

angle_rotation(optional) Symbolic or double parameter that defines the angle of rotation (in radians) of the new base. If *angle_rotation* parameter is not defined the default angle of rotation is 0.

Returns

Instance of the [Rigid_Body](#) class.

4.2.3 Member Function Documentation

4.2.3.1 function Get_Info (in *obj*, in *Info*)

Get_Info is a public function to get the properties of the [Base](#) objects.

4.2.3.2 static function Rotation_Change_Base_Matrix (in *axis*, in *angle*) [static]

Rotation_Change_Base_Matrix is a Static function that calculates a matrix to change the components of a vector in a moving base (BM) to a fixed base (BF).

Parameters

axis,: Father base axis of rotation. String that indicates which father base vector use to calculate the rotation.

angle,: Symbolic or double parameter which indicates the angle of rotation in radians.

4.2.4 Member Data Documentation

4.2.4.1 Property Axis_Labels

[3x1] symbolic array that represents the labels for the vectors of the base.

4.2.4.2 Property Base_Name

String that indicates the name of the base object.

4.2.4.3 Property M_B_E

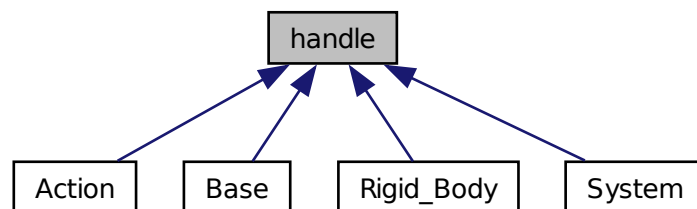
[3x3] symbolic or double array that represents the matrix to change the coordinates of a vector in the canonical base (E) to the actual base (B).

The documentation for this class was generated from the following file:

- C:/DoxygenMatlab/+Dynamic_Library/+Classes/@Base/[Base.m](#)

4.3 handle Class Reference

Inheritance diagram for handle:



The documentation for this class was generated from the following file:

- C:/DoxygenMatlab/+Dynamic_Library/+Classes/@Action/[Action.m](#)

4.4 Point Class Reference

Public Member Functions

- function [Point](#) (in Name, in varargin)
Point Class Constructor.
- function [Get_Info](#) (in obj, in Info)

Public Attributes

- Property [Point_Name](#)
- Property [Point_Coordinates_From_Canonical](#)

4.4.1 Constructor & Destructor Documentation

4.4.1.1 function Point (in *Name*, in *varargin*)

[Point](#) Class Constructor.

[Point](#) objects can be created by two different ways. A point is created in an absolute way if the point coordinates are given with respect the canonical point.

Parameters

Name,: String that defines the name of the object

point_coordinates,: [3 x 1] double or symbolic array that defines the position of the point with respect the absolute point (canonical point [0,0,0]).

The second way of creating a point is with respect a coordinate system. A coordinate system is defined by a base and a point, and the new point is given by its position with respect to the father point with the father_base components.

Parameters

Name,: String that defines the name of the object.

father_point,: Father [Point](#) object.

father_base,: Father [Base](#) object.

point_coordinates,: [3 x 1] double or symbolic array that defines the position of the point with respect the relative coordinate system (Father_Point and Father_Base).

4.4.2 Member Function Documentation

4.4.2.1 function Get_Info (in *obj*, in *Info*)

4.4.3 Member Data Documentation

4.4.3.1 Property Point_Coordinates_From_Canonical

4.4.3.2 Property Point_Name

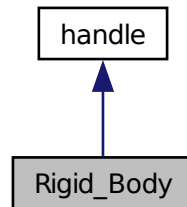
The documentation for this class was generated from the following file:

- C:/DoxygenMatlab/+Dynamic_Library/+Classes/@Point/[Point.m](#)

4.5 Rigid_Body Class Reference

Rigid Body Class for the creation of Rigid Body objects.

Inheritance diagram for Rigid_Body:



Public Member Functions

- function [Rigid_Body](#) (in Name, in varargin)
Rigid_Body Class Constructor.
- function [Get_Info](#) (in obj, in Info, in varargin)
The Get_Info method in the Rigid_Body class gets information of the rigid body object depending on the input parameter.

Public Attributes

- Property [Rigid_Body_Name](#)
Public properties for Rigid Body Class.
- Property [G_Point_From_Canonical](#)
[3x1] symbolic or double array that represents the position of the center of mass from the canonical point (in the System Class this point is named "Canonical").
- Property [Rigid_Body_Base](#)
Rigid Body base represented by the Base Class.
- Property [Mass](#)
Symbolic or double scalar that represents the mass of the Rigid Body Object.
- Property [Inertial_Tensor](#)
[3x3] symbolic or double array. This array represent the inertial tensor of the Rigid Body Object.

4.5.1 Detailed Description

Rigid Body Class for the creation of Rigid Body objects.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 function Rigid_Body (in Name, in varargin)

[Rigid_Body](#) Class Constructor.

The [Rigid_Body](#) Constructor can initialize the object by defining the parameters of a rigid body.

Parameters

name String variable that defines the name of the rigid body.

G_Point [3x1] symbolic or double array that contains the position of the center of mass of the rigid body.

Base [Base](#) object that represents the base that is connected with the rigid body.

Mass Symbolic or double scalar that represents the total mass of the rigid body.

Inertial_Tensor [3x3] symbolic or double array that represents the inertial tensor of the rigid body. If this array is a zero [3 x 3] array the object is a particle.

4.5.3 Member Function Documentation

4.5.3.1 function Get_Info (in obj, in Info, in varargin)

The Get_Info method in the [Rigid_Body](#) class gets information of the rigid body object depending on the input parameter.

Parameters

Name Every [Rigid_Body](#) object has an associated name with it. The Name option gives the name of the current rigid body object.

G_Point_From_Canonical The movement of a rigid body is given by its center of mass. This option gives the [3 x 1] symbolic or double array of the center of mass of the rigid body.

Rigid_Body_Base This option gives the base object that it is associated to the rigid body.

Inertial_Tensor This option gives the [3 x 3] array that represents inertial tensor \mathbf{I} of the rigid body.

Absolute_Velocity As the position \mathbf{r} of the rigid body center of mass is given in the canonical reference frame (and the canonical point) the velocity can be calculated as the first derivative respect to time of the position vector $\mathbf{v} = \frac{d\mathbf{r}}{dt}$. This option makes the differentiation of the position vector and returns the absolute velocity.

Absolute_Acceleration As the position \mathbf{r} of the rigid body center of mass is given in the canonical reference frame (and the canonical point) the acceleration can be calculated as the second derivative respect to time of the position vector $\mathbf{a} = \frac{d^2\mathbf{r}}{dt^2}$. This option makes the second derivative of the position vector and returns the absolute acceleration.

Absolute_Angular_Velocity The angular velocity of the rigid body is given by the components of the skew-symmetric matrix $[\mathbf{S}]^T [\mathbf{S}]$, where \mathbf{S} is the change of base matrix from the rigid body base to the canonical base. The components of the skew-symmetric matrix give the components of the angular velocity vector in the Rigid Body reference frame as $\{\boldsymbol{\Omega}_{\mathbf{R}}(\mathbf{BM})\}_{BM}$. This option calculates the vector components of the angular velocity in the Rigid Body reference frame and then returns the vector components in the canonical reference frame $\{\boldsymbol{\Omega}_{\mathbf{R}}(\mathbf{BM})\}_R$.

Virtual_Velocities In a holonomic system the real and virtual velocities can be expressed in terms of the generalized coordinates of the system. This equations for the lineal and angular velocity are given by $\mathbf{v}_i = \sum_k \frac{\partial \mathbf{v}_i}{\partial \dot{q}_k} \dot{q}_k$ and $\boldsymbol{\Omega}_i = \sum_k \frac{\partial \boldsymbol{\Omega}_i}{\partial \dot{q}_k} \dot{q}_k$, where i represents the rigid body and k the

generalized coordinates. This equations can be rewritten as $\mathbf{v}_i = \sum_k \mathbf{b}_k^i \dot{q}_k$ and $\boldsymbol{\Omega}_i = \sum_k \mathbf{c}_k^i \dot{q}_k$. The option Virtual_Velocities in the Get_Info function gives the symbolic array $[\mathbf{c}_k^i, \mathbf{b}_k^i]^T$ for the specific rigid body object i and a generalized coordinate q_k .

Kinetic_Energy This option returns the kinetic energy of the rigid body.

A_Matrix The first component (1,1) of the $[A]$ matrix of a rigid body is gonna be $-m\mathbf{b}_1^T \mathbf{b}_1 - [\mathbf{I}\mathbf{c}_1]^T \mathbf{c}_1$. In this expression m is the mass of the rigid body, \mathbf{I} is the inertial tensor of the rigid body and \mathbf{b}_1 and \mathbf{c}_1 are the virtual displacements of the rigid body with respect to the first generalized coordinate. This option returns the symbolic or double $[n \times n]$ matrix $[A]$.

d_A_Matrix This option gives the derivative matrix of matrix $[A]$.

4.5.4 Member Data Documentation

4.5.4.1 Property G_Point_From_Canonical

[3x1] symbolic or double array that represents the position of the center of mass from the canonical point (in the [System](#) Class this point is named "Canonical").

4.5.4.2 Property Inertial_Tensor

[3x3] symbolic or double array. This array represent the inertial tensor of the Rigid Body Object.

4.5.4.3 Property Mass

Symbolic or double scalar that represents the mass of the Rigid Body Object.

4.5.4.4 Property Rigid_Body_Base

Rigid Body base represented by the [Base](#) Class.

4.5.4.5 Property Rigid_Body_Name

Public properties for Rigid Body Class.

Name of Rigid Body. This property is a String class variable representing que name of the Rigid Body Object.

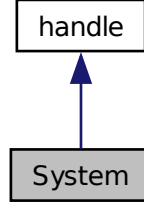
The documentation for this class was generated from the following file:

- C:/DoxygenMatlab/+Dynamic_Library/+Classes/@Rigid_Body/[Rigid_Body.m](#)

4.6 System Class Reference

[System](#) Class for the creation of [System](#) objects.

Inheritance diagram for System:



Public Member Functions

- function [System](#) (in Name)
- function [unify_system](#) (in obj)

The function `unify_system` creates the function handles for the calculation of virtual displacements of the system. This virtual displacements are used to calculate the matrix $[\mathbf{A}]$ (function `Get_A_tk_xk`), its derivative $\frac{d}{dt}[\mathbf{A}]$ (function `Get_d_A_tk_xk`) and the generalized actions τ (function `Get_tau_tk_xk` and function `Get_input_tk_xk`).

- function [unify_symbolic_system](#) (in obj)

The `unify_symbolic_system` function calculates the matrix $[\mathbf{A}]$, its derivative $\frac{d}{dt}[\mathbf{A}]$ and the generalized actions τ (the generalized actions are given by the inputs and other external actions). The system of equations is presented as $[\mathbf{A}]\ddot{\mathbf{q}} = \mathbf{b}$ where the vector \mathbf{b} has been calculated as $\mathbf{b} = -\frac{d}{dt}[\mathbf{A}]\dot{\mathbf{q}} + \tau$. As the system of equations presented is of second order the function also returns the reduced system of equations to a first order system of equations $[\mathbf{u}, \mathbf{v}, \dot{\mathbf{v}}]^T = [\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}]^T$. This way the reduced system of equations is $[[\mathbf{A}_u]\mathbf{v}, \mathbf{u}]^T = [\mathbf{b}_{uv}, \mathbf{v}]^T$ where $\mathbf{A}_u = \mathbf{A}(\mathbf{u})$ and $\mathbf{b}_{uv} = \mathbf{b}(\mathbf{u}, \mathbf{v})$.

- function [dot_A_dot_q](#) (in obj)
- function [Get_A_tk_xk](#) (in obj, in tk, in xk)

`dot_A_dot_q` function gives the $[N \times 1]$ array that represents the $[\dot{\mathbf{A}}]\dot{\mathbf{q}}$ term of the EOM.

- function [Get_d_A_tk_xk](#) (in obj, in tk, in xk)

The function `Get_A_tk_xk` calculates the value of the \mathbf{A} matrix in a x_k configuration at time t_k .

- function [Get_tau_tk_xk](#) (in obj, in tk, in xk)
- function [Get_input_tk_xk](#) (in obj, in tk, in xk, in external_vars_k)

The function `Get_input_tk_xk` calculates the value of the generalized inputs (generalized forces of the inputs) in a $[x_k, x_{external_k}]^T$ configuration at time t_k .

- function [Lagrangian_Rigth_Hand_Side](#) (in obj)

The function `Lagrangian_Rigth_Hand_Side` returns the vector $\mathbf{b} = -\frac{d}{dt}[\mathbf{A}]\dot{\mathbf{q}} + \tau$ of the $[\mathbf{A}]\ddot{\mathbf{q}} = \mathbf{b}$ system of equations.

- function [Create_New_Base](#) (in obj, in Name, in varargin)

Base introduction in `System`.

- function [Get_Base_Info](#) (in obj, in Info, in varargin)
- function [Change_Basis_Matrix](#) (in obj, in Base_1, in Base_2)
Given some vector components in a Base_1 the Change_Basis_Matrix function gives the vector components in base Base_2.
- function [Angular_Velocity](#) (in obj, in Base_1, in Base_2, in Base_Components)
Angular_Velocity function gives the angular velocity vector from Base_1 to Base_2.
- function [Create_New_Point](#) (in obj, in Name, in Coordinate_System, in Coordinates)
Point introduction in System.
- function [Get_Two_Points_Vector](#) (in obj, in Start_Point, in End_Point)
The function Get_Two_Points_Vector returns the vector components from Start_Point to End_Point in the Canonical base.
- function [Get_Two_Points_Vector_Base](#) (in obj, in Start_Point, in End_Point, in Base)
The Get_Two_Points_Vector_Base function returns the vector components from Start_Point to End_Point in any base.
- function [Get_Point_Info](#) (in obj, in Info, in varargin)
- function [Create_New_Coordinate_System](#) (in obj, in Name, in Point_Name, in Base_Name)
The Create_New_Coordinate_System function introduces a new coordinate system in the System object.
- function [Get_Coordinate_System_Info](#) (in obj, in Info, in varargin)
- function [Create_New_Rigid_Body](#) (in obj, in Name, in varargin)
Create_New_Rigid_Body is a function uses the methods of the class Rigid_Body to create a Rigid Body object.
- function [Get_Rigid_Body_Info](#) (in obj, in Info, in varargin)
- function [Get_Virtual_Velocity_q_Component](#) (in obj, in Name, in q)
The Get_Virtual_Velocity_q_Component function provides the $[\mathbf{c}_k^i, \mathbf{b}_k^i]^T$ array given by the Get_Info function of the Rigid_Body class.
- function [Get_Virtual_Velocity_q_uv_handle](#) (in obj, in Name)
The Get_Virtual_Velocity_q_uv_handle function provides the $[\mathbf{c}_k^i, \mathbf{b}_k^i]^T$ array given by the Get_Info function of the Rigid_Body class in the variables $[\mathbf{u}, \mathbf{v}]^T$ of the order reduction.
- function [Get_A_Matrix_Handle](#) (in obj, in Name)
The function Get_A_Matrix_Handle creates a $[1 \times 5]$ cell matrix for one rigid body in the system and helps the calculation of the $[A]$ values.
- function [Get_System_Virtual_Velocity_q_uv_handle](#) (in obj)
- function [Get_System_A_Matrix_Handles](#) (in obj)
The function Get_System_A_Matrix_Handles creates a $[n_{RB} \times 2]$ cell matrix, where n_{RB} is the number of rigid bodies in the system.
- function [Get_Transformation_Matrix_Handle](#) (in obj, in Name)
The function Get_Transformation_Matrix_Handle gives the function handle for the calculation of the Transformation Matrix (T) of one of the rigid bodies in the system.

- function [Get_System_Transformation_Matrix_Handle](#) (in obj)
The function `Get_System_Transformation_Matrix_Handle` returns a $[n_{RB} \times 3]$ array of cells that contains the Symbolic matrices and function handles of all of the rigid bodies transformation matrices in the system, where n_{RB} is the number of rigid bodies in the system.
- function [Get_Kinetic_Energy_Quadratic_Matrix](#) (in obj, in Name)
- function [Get_d_Kinetic_Energy_Quadratic_Matrix](#) (in obj, in Name)
The function `Get_Kinetic_Energy_Quadratic_Matrix` gets the $[A]$ matrix of a rigid body in the system.
- function [Get_System_Kinetic_Energy_Quadratic_Matrix](#) (in obj)
The function `Get_d_Kinetic_Energy_Quadratic_Matrix` gets the $[\dot{A}]$ matrix of a rigid body in the system.
- function [Get_System_d_Kinetic_Energy_Quadratic_Matrix](#) (in obj)
The function `Get_System_Kinetic_Energy_Quadratic_Matrix` gets the $[A]$ matrix of the whole system.
- function [Create_New_Action](#) (in obj, in Name, in varargin)
The `Create_New_Action` function introduces a new action in the current system.
- function [Get_Action_Info](#) (in obj, in Info, in varargin)
- function [Get_Action_in_G](#) (in obj, in Name)
- function [Get_Generalized_Action](#) (in obj, in Name)
`Get_Action_in_G` is a function that takes the Name of an [Action](#) in the [System](#) and gives the analogous [Action](#) in the center of mass G of the rigid body.
- function [Get_System_Generalized_Actions](#) (in obj)
The `Get_Generalized_Action` function gives the $[N \times 1]$ array that represents the contribution of an [Action](#) to the EOM.
- function [Create_New_Generalized_Coordinate](#) (in obj, in Name, in Symbol)
`Create_New_Generalized_Coordinate` is a function that includes in the system one generalized coordinate.
- function [Get_Generalized_Coordinates_Info](#) (in obj, in Info, in varargin)

Static Public Member Functions

- static function [order_reduction](#) (in [A](#), in [q](#), in [d_q](#))
The function `order_reduction` gets a symbolic array with \mathbf{q} and $\dot{\mathbf{q}}$ variables and gives an array with the change of variables $[\mathbf{u}, \mathbf{v}]^T = [\mathbf{q}, \dot{\mathbf{q}}]^T$.

Public Attributes

- Property [System_Name](#)
String that defines the name of the system.
- Property [q_variables](#)
 $[n \times 1]$ symbolic array that contains the generalized coordinates of the system.
- Property [Virtual_Velocity_Components_fun_handles](#)
 $[n \times 1]$ handle of the functions that calculate the virtual displacements components.

- Property [A_Matrix_Fun_Handles](#)
[n x n] handle of the functions that calculate the A matrix.
- Property [u](#)
- Property [v](#)
- Property [Transformation_Matrix_Handles](#)
- Property [A](#)
- Property [A_uv](#)
- Property [d_A](#)
- Property [d_A_uv](#)
- Property [q](#)
- Property [uv](#)
- Property [vu](#)
- Property [b](#)
- Property [b_uv](#)
- Property [System_Bases](#)
- Property [System_Points](#)
- Property [System_Coordinate_Systems](#)
- Property [System_Rigid_Bodies](#)
- Property [System_Actions](#)
- Property [System_Generalized_Coordinates](#)
- Property [System_Inputs](#)

Private Member Functions

- function [System_Create_Bases_Table](#) (in obj)
The System_Create_Bases_Table function creates a table of all of the bases in the system.
- function [System_Create_Points_Table](#) (in obj)
The System_Create_Points_Table function creates a table of all of the points in the system.
- function [System_Create_Canonical_Coordinate_System](#) (in obj)
This function creates a Canonical Coordinate [System](#) with the Canonical [Base](#) and the Canonical [Point](#).
- function [System_Create_Coordinate_System_Table](#) (in obj)
The System_Create_Coordinate_System_Table function creates a table of all of the coordinate systems in the system.
- function [System_Create_Rigid_Body_Table](#) (in obj)
The System_Create_Rigid_Body_Table function creates a table of all of the points in the system.
- function [System_Create_Canonical_action](#) (in obj)
This function creates a null action [0, 0, 0, 0, 0, 0] named Canonical.
- function [System_Create_Action_Table](#) (in obj)
The System_Create_Action_Table function creates a table of all of the external Actions in the system.
- function [System_Create_Generalized_Coordinates_Table](#) (in obj)
The System_Create_Generalized_Coordinates_Table function creates a table of all of the generalized coordinates in the system.

Static Private Member Functions

- static function [System_Create_Canonical_Base](#) ()
This function creates a Canonical base $[1, 0, 0]$, $[0, 1, 0]$, $[0, 0, 1]$.
- static function [System_Create_Canonical_Point](#) ()
This function creates a Canonical [Point](#) $[0, 0, 0]$.
- static function [System_Create_Canonical_Rigid_Body](#) ()
This function creates a Canonical Rigid Body (all default settings).

Private Attributes

- Property [System_Canonical_Base](#)
- Property [System_Canonical_Point](#)
- Property [System_Canonical_Coordinate_System](#)
- Property [System_Canonical_Rigid_Body](#)
- Property [System_Canonical_Action](#)
The function [Get_System_Kinetic_Energy_Quadratic_Matrix](#) gets the $[A]$ matrix of the whole system.
- Property [System_Canonical_Generalized_Coordinate](#)
- Property [System_Canonical_Input](#)

4.6.1 Detailed Description

[System](#) Class for the creation of [System](#) objects.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 function [System](#) (in *Name*)

4.6.3 Member Function Documentation

4.6.3.1 function [Angular_Velocity](#) (in *obj*, in *Base_1*, in *Base_2*, in *Base_Components*)

[Angular_Velocity](#) function gives the angular velocity vector from *Base_1* to *Base_2*.

The given vector components can be represented in the *Base_Components* base.

Parameters

***Base_1*,**: [Base](#) object that represent the fixed frame

***Base_2*,**: [Base](#) object that represent the moving frame

***Base_Components*,**: [Base](#) object in which the components will be given.

4.6.3.2 function Change_Basis_Matrix (in *obj*, in *Base_1*, in *Base_2*)

Given some vector components in a *Base_1* the Change_Basis_Matrix function gives the vector components in base *Base_2*.

Parameters

- Base_1*,: [Base](#) object that represent the fixed frame
- Base_2*,: [Base](#) object that represent the moving frame

4.6.3.3 function Create_New_Action (in *obj*, in *Name*, in *varargin*)

The Create_New_Action function introduces a new action in the current system.

Parameters

- obj*,: [System](#) in which to introduce a new action
- Name*,: Name of the new action
- Point*,: [Point](#) of application of the action. The point is given by its String Name and has to be introduced previously in the [System](#).
- Base*,: [Base](#) in which the action components are given. The [Base](#) is given by its String Name and has to be introduced previously in the [System](#).
- Rigid_Body*,: Rigid Body to which the action applies. The [Rigid_Body](#) is given by its String Name and has to be introduced previously in the [System](#).
- Vector*,: Components of the [Action](#). The action components can be symbolic or double and they are given by 3 torque components and 3 force components. Eg: [Mx;My;Mz;Fx;Fy;Fz].

4.6.3.4 function Create_New_Base (in *obj*, in *Name*, in *varargin*)

[Base](#) introduction in [System](#).

This function creates a [Base](#) object with the same inputs that the [Base](#) Class requires. This base is introduced in the [System](#) object.

Parameters

- obj*,: [System](#) object
- Name*
- axis_labels(optional)*,: [3x1] symbolic array that contains the labels of the 3 vectors that represents the base. If this parameter is not specified the default axis_labels will be $[x, y, z]^T$.
- father_base(optional)*,: The parent [Base](#) object defines the base from which the constructor will create a new [Base](#) object. From this parent base, a new base is generated through a simple rotation. The simple rotation can only be performed in the direction indicated by the right-hand rule for each of the vectors of the parent base. If father_base parameter is not defined, the default parent base is [1, 0, 0], [0, 1, 0], and [0, 0, 1].
- axis_rotation(optional)*,: The simple rotation can only be performed in the direction indicated by the right-hand rule for each of the vectors of the parent base. The axis_rotation parameter defines the rotation axis with a string "1", "2" or "3" defining the parent basis vector to use. If axis_rotation parameter is not defined, the default axis of rotation "1".
- angle_rotation(optional)*,: Symbolic or double parameter that defines the angle of rotation (in radians) of the new base. If angle_rotation parameter is not defined the default angle of rotation is 0.

4.6.3.5 function Create_New_Coordinate_System (in *obj*, in *Name*, in *Point_Name*, in *Base_Name*)

The Create_New_Coordinate_System function introduces a new coordinate system in the [System](#) object. This Coordinate [System](#) is given by a [Point](#) and a [Base](#) object.

Parameters

Name,: Name given to the new coordinate system.

Point_Name,: [Point](#) Name (String) of the [Point](#) used to create the new coordinate system.

Base_Name,: [Base](#) Name (String) of the [Base](#) used to create the new coordinate system.

4.6.3.6 function Create_New_Generalized_Coordinate (in *obj*, in *Name*, in *Symbol*)

Create_New_Generalized_Coordinate is a function that includes in the system one generalized coordinate.

Parameters

obj,: [System](#) object in which to include the new generalized coordinate.

Name,: Generalized coordinate name (string).

Symbol,: Generalized coordinate symbol (symbolic variable)

4.6.3.7 function Create_New_Point (in *obj*, in *Name*, in *Coordinate_System*, in *Coordinates*)

[Point](#) introduction in [System](#).

This function creates a [Point](#) object with the same inputs that the [Point](#) Class requires. This base is introduced in the [System](#) object.

Parameters

Name,: String that defines the name of the object

point_coordinates,: [3 x 1] double or symbolic array that defines the position of the point with respect to the absolute point (canonical point [0,0,0]).

The second way of creating a point is with respect to a coordinate system. A coordinate system is defined by a base and a point, and the new point is given by its position with respect to the father point with the father_base components.

Parameters

Name,: String that defines the name of the object.

father_point,: Father [Point](#) (String).

father_base,: Father [Base](#) (String).

point_coordinates,: [3 x 1] double or symbolic array that defines the position of the point with respect to the relative coordinate system (Father_Point and Father_Base).

4.6.3.8 function Create_New_Rigid_Body (in *obj*, in *Name*, in *varargin*)

Create_New_Rigid_Body is a function uses the methods of the class [Rigid_Body](#) to create a Rigid Body object.

Parameters

name,: String that defines the name of the rigid body.

G_Point,: Center of mass of the rigid body from the canonical point OG. Symbolic or double [3 x 1] array.

Base,: [Base](#) object that defines the kinematic base linked with the rigid body.

Mass,: Symbolic or double scalar that represents the total mass of the rigid body.

Inertial_Tensor,: [3 x 3] symbolic or double variable that represents the inertial tensor of the rigid body. A zero [3 x 3] array represents a particle.

4.6.3.9 function dot_A_dot_q (in *obj*)

4.6.3.10 function Get_A_Matrix_Handle (in *obj*, in *Name*)

The function Get_A_Matrix_Handle creates a [1 × 5] cell matrix for one rigid body in the system and helps the calculation of the [A] values.

The cell arrays columns are: name of the rigid body (string), mass of the rigid body (double number), inertial tensor of the rigid body ([3 × 3] double array), [n × n] cell array used to get the [A] matrix and a [n × n] cell array used to get the [Ȧ] matrix. The value of *n* is the number of generalized coordinates in the system (the degrees of freedom in a holonomic system).

Parameters

Name Name of the rigid body (string)

Each of the components of the [n × n] cell array contains the function handles to get the numeric values of the [A] and [Ȧ] matrices. For example, the first component (1,1) of the [A] matrix of a rigid body is gonna be $-mb_1^T b_1 - [Ic_1]^T c_1$. In this expression *m* is the mass of the rigid body, **I** is the inertial tensor of the rigid body and **b**₁ and **c**₁ are the virtual displacements of the rigid body with respect to the first generalized coordinate. The [n × n] cell array gives the handles to calculate the virtual displacements **b**₁ and **c**₁ and in which order the product should be done. With this handles, the mass, and the inertial tensor, the numerical (1,1) component of the [A] matrix can be calculated. Analogous methodology is used for the other components and also for the [Ȧ] matrix aswell.

4.6.3.11 function Get_A_tk_xk (in *obj*, in *tk*, in *xk*)

dot_A_dot_q function gives the [N x 1] array that represents the [Ȧ]q̇ term of the EOM.

This function is used for the calculation of the RHS terms.

Parameters

obj [System](#)

4.6.3.12 function `Get_Action_in_G` (in *obj*, in *Name*)

4.6.3.13 function `Get_Action_Info` (in *obj*, in *Info*, in *varargin*)

4.6.3.14 function `Get_Base_Info` (in *obj*, in *Info*, in *varargin*)

4.6.3.15 function `Get_Coordinate_System_Info` (in *obj*, in *Info*, in *varargin*)

4.6.3.16 function `Get_d_A_tk_xk` (in *obj*, in *tk*, in *xk*)

The function `Get_A_tk_xk` calculates the value of the A matrix in a x_k configuration at time t_k .

Parameters

obj [System](#)

tk time t_k

xk configuration x_k

4.6.3.17 function `Get_d_Kinetic_Energy_Quadratic_Matrix` (in *obj*, in *Name*)

The function `Get_Kinetic_Energy_Quadratic_Matrix` gets the $[A]$ matrix of a rigid body in the system.

This function uses the `Get_Info` function of the [Rigid_Body](#) class with the `A_Matrix` option to get the A matrix.

Parameters

Name Name of the rigid body.

4.6.3.18 function `Get_Generalized_Action` (in *obj*, in *Name*)

`Get_Action_in_G` is a function that takes the Name of an [Action](#) in the [System](#) and gives the analogous [Action](#) in the center of mass G of the rigid body.

Parameters

Name Name of the action.

4.6.3.19 function `Get_Generalized_Coordinates_Info` (in *obj*, in *Info*, in *varargin*)

4.6.3.20 function `Get_input_tk_xk` (in *obj*, in *tk*, in *xk*, in *external_vars_k*)

The function `Get_input_tk_xk` calculates the value of the generalized inputs (generalized forces of the inputs) in a $[x_k, x_{external_k}]^T$ configuration at time t_k .

Parameters

obj [System](#)

tk time t_k

xk configuration x_k

4.6.3.21 function Get_Kinetic_Energy_Quadratic_Matrix (in *obj*, in *Name*)

4.6.3.22 function Get_Point_Info (in *obj*, in *Info*, in *varargin*)

4.6.3.23 function Get_Rigid_Body_Info (in *obj*, in *Info*, in *varargin*)

4.6.3.24 function Get_System_A_Matrix_Handles (in *obj*)

The function `Get_System_A_Matrix_Handles` creates a $[n_{RB} \times 2]$ cell matrix, where n_{RB} is the number of rigid bodies in the system.

This cell arrays give the function handles and parameters to calculate the $[A]$ and $[\dot{A}]$ matrices. This matrices represent the contribution of the inertial forces to the EOM.

Parameters

obj [System](#)

4.6.3.25 function Get_System_d_Kinetic_Energy_Quadratic_Matrix (in *obj*)

The function `Get_System_Kinetic_Energy_Quadratic_Matrix` gets the $[A]$ matrix of the whole system.

This function uses the `Get_Kinetic_Energy_Quadratic_Matrix` function and the system rigid body table to merge all the individual matrices.

4.6.3.26 function Get_System_Generalized_Actions (in *obj*)

The `Get_Generalized_Action` function gives the $[N \times 1]$ array that represents the contribution of an [Action](#) to the EOM.

Generally this components are known as generalized forces.

Parameters

Name Name of the [Action](#) The `Get_System_Generalized_Actions` function gives the $[N \times 1]$ array that represents the contribution of all Actions of the [System](#) to the EOM. Generally this components are known as generalized forces.

obj [System](#)

4.6.3.27 function Get_System_Kinetic_Energy_Quadratic_Matrix (in *obj*)

The function `Get_d_Kinetic_Energy_Quadratic_Matrix` gets the $[\dot{A}]$ matrix of a rigid body in the system.

This function uses the `Get_Info` function of the [Rigid_Body](#) class with the `d_A_Matrix` option to get the A matrix.

Parameters

Name Name of the rigid body.

4.6.3.28 function Get_System_Transformation_Matrix_Handle (in *obj*)

The function `Get_System_Transformation_Matrix_Handle` returns a $[n_{RB} \times 3]$ array of cells that contains the Symbolic matrices and function handles of all of the rigid bodies transformation matrices in the system, where n_{RB} is the number of rigid bodies in the system.

The first column of the cells represent the name (string) of each rigid body, the second contains the Symbolic Transformation Matrix and the third one keeps the function handle for the numerical calculation of the transformation matrix.

4.6.3.29 function Get_System_Virtual_Velocity_q_uv_handle (in *obj*)

4.6.3.30 function Get_tau_tk_xk (in *obj*, in *tk*, in *xk*)

4.6.3.31 function Get_Transformation_Matrix_Handle (in *obj*, in *Name*)

The function `Get_Transformation_Matrix_Handle` gives the function handle for the calculation of the Transformation Matrix (T) of one of the rigid bodies in the system.

This transformation matrix is given in the reduced order variables $[\mathbf{u}, \mathbf{v}]^T$.

Parameters

Name

Return values

T_uv Symbolic Transformation Matrix (T).

T_fun_handle Function handle of the Transformation Matrix (T).

4.6.3.32 function Get_Two_Points_Vector (in *obj*, in *Start_Point*, in *End_Point*)

The function `Get_Two_Points_Vector` returns the vector components from *Start_Point* to *End_Point* in the Canonical base.

Parameters

Start_Point The initial point.

End_Point The end point.

Return values

out $[3 \times 1]$ position vector

4.6.3.33 function Get_Two_Points_Vector_Base (in *obj*, in *Start_Point*, in *End_Point*, in *Base*)

The `Get_Two_Points_Vector_Base` function returns the vector components from *Start_Point* to *End_Point* in any base.

Parameters

Start_Point The initial point.

End_Point The end point.

Base Components base (String).

Return values

out $[3 \times 1]$ position vector

4.6.3.34 function Get_Virtual_Velocity_q_Component (in obj, in Name, in q)

The Get_Virtual_Velocity_q_Component function provides the $[c_k^i, b_k^i]^T$ array given by the Get_Info function of the [Rigid_Body](#) class.

Parameters

Name Name of the rigid body.

q Generalized coordinate

4.6.3.35 function Get_Virtual_Velocity_q_uv_handle (in obj, in Name)

The Get_Virtual_Velocity_q_uv_handle function provides the $[c_k^i, b_k^i]^T$ array given by the Get_Info function of the [Rigid_Body](#) class in the variables $[u, v]^T$ of the order reduction.

This function saves for each of the generalized coordinates in the system a function in the folder fun_handles in the work directory.

Parameters

Name Name of the rigid body.

4.6.3.36 function Lagrangian_Rigth_Hand_Side (in obj)

The function Lagrangian_Rigth_Hand_Side returns the vector $b = -\frac{d}{dt} [A] \dot{q} + \tau$ of the $[A] \ddot{q} = b$ system of equations.

4.6.3.37 static function order_reduction (in A, in q, in d_q) [static]

The function order_reduction gets a symbolic array with q and \dot{q} variables and gives an array with the change of variables $[u, v]^T = [q, \dot{q}]^T$.

4.6.3.38 function System_Create_Action_Table (in obj) [private]

The System_Create_Action_Table function creates a table of all of the external Actions in the system.

The table has 3 columns: Action_Name(string variable), [Rigid_Body](#) ([Rigid_Body](#) class variable) and Action(Action class variable). Each time the user introduces a new [Action](#) in the system the table is updated and a new [Action](#) is added.

4.6.3.39 function System_Create_Bases_Table (in *obj*) [private]

The System_Create_Bases_Table function creates a table of all of the bases in the system.

The table has 2 columns: Name(string variable) and Base (Base class variable). Each time the user introduces a base in the system the table is updated and a new base is added.

4.6.3.40 function System_Create_Canonical_action (in *obj*) [private]

This function creates a null action [0, 0, 0, 0, 0, 0] named Canonical.

This Canonical Action is needed for the first time creation of the Action table.

4.6.3.41 static function System_Create_Canonical_Base () [static, private]

This function creates a Canonical base [1, 0, 0] , [0, 1, 0] , [0, 0, 1].

This Canonical Base is needed for the first time creation of the Base table.

4.6.3.42 function System_Create_Canonical_Coordinate_System (in *obj*) [private]

This function creates a Canonical Coordinate System with the Canonical Base and the Canonical Point.

This Canonical Coordinate System is needed for the first time creation of the Coordinate System table.

4.6.3.43 static function System_Create_Canonical_Point () [static, private]

This function creates a Canonical Point [0, 0, 0].

This Canonical Point is needed for the first time creation of the Points table.

4.6.3.44 static function System_Create_Canonical_Rigid_Body () [static, private]

This function creates a Canonical Rigid Body (all default settings).

This Canonical Rigid Body is needed for the first time creation of the Rigid Bodies table.

4.6.3.45 function System_Create_Coordinate_System_Table (in *obj*) [private]

The System_Create_Coordinate_System_Table function creates a table of all of the coordinate systems in the system.

The table has 2 columns: Point_Name(string variable) and Base_Name (string variable). Each time the user introduces a new coordinate system in the system the table is updated and a new coordinate system is added.

4.6.3.46 function System_Create_Generalized_Coordinates_Table (in *obj*) [private]

The System_Create_Generalized_Coordinates_Table function creates a table of all of the generalized coordinates in the system.

The table has 2 columns: Generalized_Coordinate_Name(string variable) and Generalized_Coordinate_Symbol(symbolic class variable). Each time the user introduces a new generalized coordinate in the system the table is updated and a new generalized coordinate is added.

4.6.3.47 function System_Create_Points_Table (in *obj*) [private]

The System_Create_Points_Table function creates a table of all of the points in the system.

The table has 2 columns: Name(string variable) and [Point](#) ([Point](#) class variable). Each time the user introduces a point in the system the table is updated and a new point is added.

4.6.3.48 function System_Create_Rigid_Body_Table (in *obj*) [private]

The System_Create_Rigid_Body_Table function creates a table of all of the points in the system.

The table has 4 columns: Rigid_Body_Name(string variable), Point_Name (string variable), Base_Name(string variable) and Rigid_Body(Rigid_Body class variable) . Each time the user introduces a rigid body in the system the table is updated and a new rigid body is added.

4.6.3.49 function unify_symbolic_system (in *obj*)

The unify_symbolic_system function calculates the matrix $[\mathbf{A}]$, its derivative $\frac{d}{dt} [\mathbf{A}]$ and the generalized actions τ (the generalized actions are given by the inputs and other external actions). The system of equations is presented as $[\mathbf{A}] \ddot{\mathbf{q}} = \mathbf{b}$ where the vector \mathbf{b} has been calculated as $\mathbf{b} = -\frac{d}{dt} [\mathbf{A}] \dot{\mathbf{q}} + \tau$. As the system of equations presented is of second order the function also returns the reduced system of equations to a first order system of equations $[\mathbf{u}, \mathbf{v}, \dot{\mathbf{v}}]^T = [\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}]^T$. This way the reduced system of equations is $[[\mathbf{A}_u] \mathbf{v}, \mathbf{u}]^T = [\mathbf{b}_{uv}, \mathbf{v}]^T$ where $\mathbf{A}_u = \mathbf{A}(\mathbf{u})$ and $\mathbf{b}_{uv} = \mathbf{b}(\mathbf{u}, \mathbf{v})$.

4.6.3.50 function unify_system (in *obj*)

The function unify_system creates the function handles for the calculation of virtual displacements of the system. This virtual displacements are used to calculate the matrix $[\mathbf{A}]$ (function `Get_A_tk_xk`), its derivative $\frac{d}{dt} [\mathbf{A}]$ (function `Get_d_A_tk_xk`) and the generalized actions τ (function `Get_tau_tk_xk` and function `Get_input_tk_xk`).

4.6.4 Member Data Documentation

4.6.4.1 Property A

4.6.4.2 Property A_Matrix_Fun_Handles

[n x n] handle of the functions that calculate the A matrix.

4.6.4.3 Property A_uv**4.6.4.4 Property b****4.6.4.5 Property b_uv****4.6.4.6 Property d_A****4.6.4.7 Property d_A_uv****4.6.4.8 Property q****4.6.4.9 Property q_variables**

[n x 1] symbolic array that contains the generalized coordinates of the system.

4.6.4.10 Property System_Actions**4.6.4.11 Property System_Bases****4.6.4.12 Property System_Canonical_Action [private]**

The function Get_System_Kinetic_Energy_Quadratic_Matrix gets the $[\dot{A}]$ matrix of the whole system.

This function uses the Get_d_Kinetic_Energy_Quadratic_Matrix function and the system rigid body table to merge all the individual matrices.

4.6.4.13 Property System_Canonical_Base [private]**4.6.4.14 Property System_Canonical_Coordinate_System [private]****4.6.4.15 Property System_Canonical_Generalized_Coordinate [private]****4.6.4.16 Property System_Canonical_Input [private]****4.6.4.17 Property System_Canonical_Point [private]****4.6.4.18 Property System_Canonical_Rigid_Body [private]****4.6.4.19 Property System_Coordinate_Systems****4.6.4.20 Property System_Generalized_Coordinates****4.6.4.21 Property System_Inputs****4.6.4.22 Property System_Name**

String that defines the name of the system.

4.6.4.23 Property System_Points**4.6.4.24 Property System_Rigid_Bodies****4.6.4.25 Property Transformation_Matrix_Handles****4.6.4.26 Property u****4.6.4.27 Property uv****4.6.4.28 Property v****4.6.4.29 Property Virtual_Velocity_Components_fun_handles**

[n x 1] handle of the functions that calculate the virtual displacements components.

4.6.4.30 Property vu

The documentation for this class was generated from the following file:

- C:/DoxygenMatlab/+Dynamic_Library/[System.m](#)

Chapter 5

File Documentation

5.1 C:/DoxygenMatlab/+Dynamic_Library/+Classes/@Action/Action.m File Reference

Classes

- class [Action](#)

5.2 C:/DoxygenMatlab/+Dynamic_Library/+Classes/@Base/Base.m File Reference

[Base](#) Class description.

Classes

- class [Base](#)
[Base](#) Class for the creation of orthonormal vectorial bases.

5.2.1 Detailed Description

[Base](#) Class description.

5.3 C:/DoxygenMatlab/+Dynamic_Library/+Classes/@Point/Point.m File Reference

Classes

- class [Point](#)

5.4 C:/DoxygenMatlab/+Dynamic_Library/+Classes/@Rigid_Body/Rigid_Body.m File Reference

Rigid Body Class description.

Classes

- class [Rigid_Body](#)
Rigid Body Class for the creation of Rigid Body objects.

5.4.1 Detailed Description

Rigid Body Class description.

5.5 C:/DoxygenMatlab/+Dynamic_Library/System.m File Reference

[System](#) Class description.

Classes

- class [System](#)
[System](#) Class for the creation of [System](#) objects.

5.5.1 Detailed Description

[System](#) Class description.

Index

- A
 - System, [29](#)
- A_Matrix_Fun_Handles
 - System, [29](#)
- A_uv
 - System, [29](#)
- Action, [7](#)
 - Action, [8](#)
 - Action_Base, [8](#)
 - Action_Name, [8](#)
 - Action_Point, [8](#)
 - Action_Vector, [8](#)
 - Get_Info, [8](#)
- Action_Base
 - Action, [8](#)
- Action_Name
 - Action, [8](#)
- Action_Point
 - Action, [8](#)
- Action_Vector
 - Action, [8](#)
- Angular_Velocity
 - System, [20](#)
- Axis_Labels
 - Base, [11](#)
- b
 - System, [30](#)
- b_uv
 - System, [30](#)
- Base, [8](#)
 - Axis_Labels, [11](#)
 - Base, [10](#)
 - Base_Name, [11](#)
 - Get_Info, [10](#)
 - M_B_E, [11](#)
 - Rotation_Change_Base_Matrix, [10](#)
- Base_Name
 - Base, [11](#)
- C:/DoxygenMatlab/+Dynamic_Library/System.m,
[34](#)
- Change_Basis_Matrix
 - System, [20](#)
- Create_New_Action
 - System, [21](#)
- Create_New_Base
 - System, [21](#)
- Create_New_Coordinate_System
 - System, [21](#)
- Create_New_Generalized_Coordinate
 - System, [22](#)
- Create_New_Point
 - System, [22](#)
- Create_New_Rigid_Body
 - System, [22](#)
- d_A
 - System, [30](#)
- d_A_uv
 - System, [30](#)
- dot_A_dot_q
 - System, [23](#)
- G_Point_From_Canonical
 - Rigid_Body, [15](#)
- Get_A_Matrix_Handle
 - System, [23](#)
- Get_A_tk_xk
 - System, [23](#)
- Get_Action_in_G
 - System, [23](#)
- Get_Action_Info
 - System, [24](#)
- Get_Base_Info
 - System, [24](#)
- Get_Coordinate_System_Info
 - System, [24](#)
- Get_d_A_tk_xk
 - System, [24](#)
- Get_d_Kinetic_Energy_Quadratic_Matrix
 - System, [24](#)
- Get_Generalized_Action
 - System, [24](#)
- Get_Generalized_Coordinates_Info
 - System, [24](#)
- Get_Info
 - Action, [8](#)
 - Base, [10](#)
 - Point, [12](#)

- Rigid_Body, [14](#)
- Get_input_tk_xk
 - System, [24](#)
- Get_Kinetic_Energy_Quadratic_Matrix
 - System, [24](#)
- Get_Point_Info
 - System, [25](#)
- Get_Rigid_Body_Info
 - System, [25](#)
- Get_System_A_Matrix_Handles
 - System, [25](#)
- Get_System_d_Kinetic_Energy_Quadratic_Matrix
 - System, [25](#)
- Get_System_Generalized_Actions
 - System, [25](#)
- Get_System_Kinetic_Energy_Quadratic_Matrix
 - System, [25](#)
- Get_System_Transformation_Matrix_Handle
 - System, [25](#)
- Get_System_Virtual_Velocity_q_uv_handle
 - System, [26](#)
- Get_tau_tk_xk
 - System, [26](#)
- Get_Transformation_Matrix_Handle
 - System, [26](#)
- Get_Two_Points_Vector
 - System, [26](#)
- Get_Two_Points_Vector_Base
 - System, [26](#)
- Get_Virtual_Velocity_q_Component
 - System, [27](#)
- Get_Virtual_Velocity_q_uv_handle
 - System, [27](#)
- handle, [11](#)
- Inertial_Tensor
 - Rigid_Body, [15](#)
- Lagrangian_Rigth_Hand_Side
 - System, [27](#)
- M_B_E
 - Base, [11](#)
- Mass
 - Rigid_Body, [15](#)
- order_reduction
 - System, [27](#)
- Point, [11](#)
 - Get_Info, [12](#)
 - Point, [12](#)
 - Point_Coordinates_From_Canonical, [12](#)
 - Point_Name, [12](#)
- Point_Coordinates_From_Canonical
 - Point, [12](#)
- Point_Name
 - Point, [12](#)
- q
 - System, [30](#)
- q_variables
 - System, [30](#)
- Rigid_Body, [12](#)
 - G_Point_From_Canonical, [15](#)
 - Get_Info, [14](#)
 - Inertial_Tensor, [15](#)
 - Mass, [15](#)
 - Rigid_Body, [14](#)
 - Rigid_Body_Base, [15](#)
 - Rigid_Body_Name, [15](#)
 - Rigid_Body, [14](#)
- Rigid_Body_Base
 - Rigid_Body, [15](#)
- Rigid_Body_Name
 - Rigid_Body, [15](#)
- Rotation_Change_Base_Matrix
 - Base, [10](#)
- System, [15](#)
 - A, [29](#)
 - A_Matrix_Fun_Handles, [29](#)
 - A_uv, [29](#)
 - Angular_Velocity, [20](#)
 - b, [30](#)
 - b_uv, [30](#)
 - Change_Basis_Matrix, [20](#)
 - Create_New_Action, [21](#)
 - Create_New_Base, [21](#)
 - Create_New_Coordinate_System, [21](#)
 - Create_New_Generalized_Coordinate, [22](#)
 - Create_New_Point, [22](#)
 - Create_New_Rigid_Body, [22](#)
 - d_A, [30](#)
 - d_A_uv, [30](#)
 - dot_A_dot_q, [23](#)
 - Get_A_Matrix_Handle, [23](#)
 - Get_A_tk_xk, [23](#)
 - Get_Action_in_G, [23](#)
 - Get_Action_Info, [24](#)
 - Get_Base_Info, [24](#)
 - Get_Coordinate_System_Info, [24](#)
 - Get_d_A_tk_xk, [24](#)
 - Get_d_Kinetic_Energy_Quadratic_Matrix, [24](#)
 - Get_Generalized_Action, [24](#)
 - Get_Generalized_Coordinates_Info, [24](#)
 - Get_input_tk_xk, [24](#)

- Get_Kinetic_Energy_Quadratic_Matrix, 24
- Get_Point_Info, 25
- Get_Rigid_Body_Info, 25
- Get_System_A_Matrix_Handles, 25
- Get_System_d_Kinetic_Energy_Quadratic_Matrix, 25
- Get_System_Generalized_Actions, 25
- Get_System_Kinetic_Energy_Quadratic_Matrix, 25
- Get_System_Transformation_Matrix_Handle, 25
- Get_System_Virtual_Velocity_q_uv_handle, 26
- Get_tau_tk_xk, 26
- Get_Transformation_Matrix_Handle, 26
- Get_Two_Points_Vector, 26
- Get_Two_Points_Vector_Base, 26
- Get_Virtual_Velocity_q_Component, 27
- Get_Virtual_Velocity_q_uv_handle, 27
- Lagrangian_Righ_Hand_Side, 27
- order_reduction, 27
- q, 30
- q_variables, 30
- System, 20
- System_Actions, 30
- System_Bases, 30
- System_Canonical_Action, 30
- System_Canonical_Base, 30
- System_Canonical_Coordinate_System, 30
- System_Canonical_Generalized_Coordinate, 30
- System_Canonical_Input, 30
- System_Canonical_Point, 30
- System_Canonical_Rigid_Body, 30
- System_Coordinate_Systems, 30
- System_Create_Action_Table, 27
- System_Create_Bases_Table, 27
- System_Create_Canonical_action, 28
- System_Create_Canonical_Base, 28
- System_Create_Canonical_Coordinate_System, 28
- System_Create_Canonical_Rigid_Body, 28
- System_Create_Coordinate_System_Table, 28
- System_Create_Generalized_Coordinates_Table, 28
- System_Create_Points_Table, 29
- System_Create_Rigid_Body_Table, 29
- System_Generalized_Coordinates, 30
- System_Inputs, 30
- System_Name, 30
- System_Points, 30
- System_Rigid_Bodies, 31
- Transformation_Matrix_Handles, 31
- u, 31
- unify_symbolic_system, 29
- unify_system, 29
- uv, 31
- v, 31
- Virtual_Velocity_Components_fun_handles, 31
- vu, 31
- System_Actions
 - System, 30
- System_Bases
 - System, 30
- System_Canonical_Action
 - System, 30
- System_Canonical_Base
 - System, 30
- System_Canonical_Coordinate_System
 - System, 30
- System_Canonical_Generalized_Coordinate
 - System, 30
- System_Canonical_Input
 - System, 30
- System_Canonical_Point
 - System, 30
- System_Canonical_Rigid_Body
 - System, 30
- System_Coordinate_Systems
 - System, 30
- System_Create_Action_Table
 - System, 27
- System_Create_Bases_Table
 - System, 27
- System_Create_Canonical_action
 - System, 28
- System_Create_Canonical_Base
 - System, 28
- System_Create_Canonical_Coordinate_System
 - System, 28
- System_Create_Canonical_Point
 - System, 28
- System_Create_Canonical_Rigid_Body
 - System, 28
- System_Create_Coordinate_System_Table
 - System, 28
- System_Create_Generalized_Coordinates_Table
 - System, 28
- System_Create_Points_Table
 - System, 29
- System_Create_Rigid_Body_Table
 - System, 29
- System_Generalized_Coordinates
 - System, 30
- System_Inputs
 - System, 30

System_Name
 System, [30](#)
System_Points
 System, [30](#)
System_Rigid_Bodies
 System, [31](#)

Transformation_Matrix_Handles
 System, [31](#)

u
 System, [31](#)
unify_symbolic_system
 System, [29](#)
unify_system
 System, [29](#)
uv
 System, [31](#)

v
 System, [31](#)
Virtual_Velocity_Components_fun_handles
 System, [31](#)
vu
 System, [31](#)