

IK SOLVER

Generated by Doxygen 1.8.17

1 6 DOF Manipulator - Inverse Kinematics Solver	1
1.1 Project Description	1
1.1.1 Overview	1
1.1.2 Method	1
2 Namespace Index	3
2.1 Namespace List	3
3 Hierarchical Index	5
3.1 Class Hierarchy	5
4 Class Index	7
4.1 Class List	7
5 File Index	9
5.1 File List	9
6 Namespace Documentation	11
6.1 simulation Namespace Reference	11
6.1.1 Variable Documentation	11
6.1.1.1 abc1	12
6.1.1.2 abc2	12
6.1.1.3 abc3	12
6.1.1.4 abc4	12
6.1.1.5 abc5	12
6.1.1.6 abc6	12
6.1.1.7 category	12
6.1.1.8 dh_a	12
6.1.1.9 dh_alpha	13
6.1.1.10 dh_d	13
6.1.1.11 dh_params	13
6.1.1.12 dh_theta	13
6.1.1.13 frame1	13
6.1.1.14 frame2	13
6.1.1.15 frame3	13
6.1.1.16 frame4	13
6.1.1.17 frame5	14
6.1.1.18 frame6	14
6.1.1.19 frames	14
6.1.1.20 motion	14
6.1.1.21 robot	14
6.1.1.22 trajectory	14
6.1.1.23 xyz1	14
6.1.1.24 xyz2	14

6.1.1.25 xyz3	15
6.1.1.26 xyz4	15
6.1.1.27 xyz5	15
6.1.1.28 xyz6	15
7 Class Documentation	17
7.1 ForwardKinematics Class Reference	17
7.1.1 Detailed Description	18
7.1.2 Member Function Documentation	18
7.1.2.1 calculate_TF()	18
7.1.2.2 solve_fk()	18
7.1.3 Member Data Documentation	19
7.1.3.1 euler_x	19
7.1.3.2 euler_y	19
7.1.3.3 euler_z	19
7.2 InverseKinematics Class Reference	19
7.2.1 Detailed Description	20
7.2.2 Constructor & Destructor Documentation	20
7.2.2.1 InverseKinematics()	20
7.2.3 Member Function Documentation	21
7.2.3.1 get_eff_position()	21
7.2.3.2 get_eff_rotation()	21
7.2.3.3 set_eff_position()	21
7.2.3.4 set_eff_rotation()	21
7.2.3.5 solve_ik()	22
7.3 RobotParameters Class Reference	22
7.3.1 Detailed Description	23
7.3.2 Constructor & Destructor Documentation	23
7.3.2.1 RobotParameters()	23
7.3.3 Member Function Documentation	23
7.3.3.1 get_dh_parameters()	23
7.3.3.2 get_robot_angles()	24
7.3.3.3 set_dh_parameters()	24
7.3.3.4 set_robot_angles()	24
7.3.4 Member Data Documentation	24
7.3.4.1 robot_name	24
8 File Documentation	25
8.1 app/CMakeLists.txt File Reference	25
8.1.1 Function Documentation	25
8.1.1.1 add_executable()	25
8.1.1.2 find_package()	25
8.2 test/CMakeLists.txt File Reference	25

8.2.1 Function Documentation	26
8.2.1.1 set()	26
8.3 app/forward_kinematics.cpp File Reference	26
8.3.1 Detailed Description	26
8.4 app/inverse_kinematics.cpp File Reference	27
8.5 app/main.cpp File Reference	27
8.5.1 Detailed Description	28
8.5.2 Function Documentation	29
8.5.2.1 main()	29
8.6 test/main.cpp File Reference	29
8.6.1 Detailed Description	30
8.6.2 Function Documentation	30
8.6.2.1 main()	30
8.7 app/robot_parameters.cpp File Reference	30
8.7.1 Detailed Description	31
8.8 app/simulation.py File Reference	31
8.9 docs/introduction.txt File Reference	32
8.10 include/forward_kinematics.hpp File Reference	32
8.10.1 Detailed Description	33
8.11 include/inverse_kinematics.hpp File Reference	33
8.11.1 Detailed Description	34
8.12 include/robot_parameters.hpp File Reference	35
8.12.1 Detailed Description	35
8.13 test/code_test.cpp File Reference	36
8.13.1 Detailed Description	37
8.13.2 Function Documentation	37
8.13.2.1 TEST() [1/8]	38
8.13.2.2 TEST() [2/8]	38
8.13.2.3 TEST() [3/8]	38
8.13.2.4 TEST() [4/8]	38
8.13.2.5 TEST() [5/8]	38
8.13.2.6 TEST() [6/8]	39
8.13.2.7 TEST() [7/8]	39
8.13.2.8 TEST() [8/8]	39

Chapter 1

6 DOF Manipulator - Inverse Kinematics Solver

1.1 Project Description

1.1.1 Overview

The aim of this project is to design and develop an inverse kinematics solver for a 6-axis Manipulator at ACME Robotics. This implementation is based on the Kuka-KR 5 Manipulator, however it can provide a solution for any 6 axis manipulator with a spherical wrist configuration. Our software will compute and simulate a trajectory based on the path coordinates provided and will be integrated as a module into a future project over at ACME Robotics.

1.1.2 Method

Our implementation includes three methods, two for solving inverse and forwards kinematics for the manipulator, and one method to store the attributes of the specified robot arm. The solution for inverse kinematics will be calculated and stored as joint angles for each path coordinate provided. The forward kinematics solver will calculate end effector position based on the joint values and robot parameters provided. Denavit Hartenberg representation will be employed for both the inverse and forward kinematics solvers. A two sprint Agile Iterative Process approach and test driven development style is utilized in the making of this project.

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

simulation	11
--------------------------------------	----

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

RobotParameters	22
ForwardKinematics	17
InverseKinematics	19

Chapter 4

Class Index

4.1 Class List

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

ForwardKinematics	
Definition of the Forward Kinematics Class	17
InverseKinematics	
Definition of Inverse Kinematics Class	19
RobotParameters	
Definition of the Robot Parameter Class	22

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

app/forward_kinematics.cpp	
Program to define the Methods of Forward Kinematics Class	26
app/inverse_kinematics.cpp	27
app/main.cpp	
Program to execute the inverse kinematics and forward kinematics	27
app/robot_parameters.cpp	
Program to define the Methods of Robot Parameters Class	30
app/simulation.py	31
include/forward_kinematics.hpp	
Definition of Forward Kinematics class and Declaration of its Methods	32
include/inverse_kinematics.hpp	33
include/robot_parameters.hpp	
Definition of Robot Parameters class and Declaration of its Methods	35
test/code_test.cpp	
Program to perform unit testing	36
test/main.cpp	
Program to use google test for unit testing	29

Chapter 6

Namespace Documentation

6.1 simulation Namespace Reference

Variables

- [category](#)
- list [dh_d](#) = [0.25, 0.15, 0.2, 0, 0, 0.1]
- list [dh_a](#) = [0, 0, 0, 0, 0, 0]
- list [dh_alpha](#) = [-1.5708, 1.5708, 0, -1.5708, 1.5708, 0]
- list [dh_theta](#) = [0, 0, 0, 0, 0, 0]
- [dh_params](#) = np.stack(([dh_d](#),[dh_a](#),[dh_alpha](#),[dh_theta](#)), axis=-1)
- [robot](#) = RobotSerial([dh_params](#))
- [abc1](#) = np.array([[-0.266], [0.031], [0.304]])
- [abc2](#) = np.array([[-0.232], [0.192], [0.357]])
- [abc3](#) = np.array([[-0.076], [0.225], [0.461]])
- [abc4](#) = np.array([[-0.014], [0.262], [0.262]])
- [abc5](#) = np.array([[-0.221], [0.020], [0.482]])
- [abc6](#) = np.array([[-0.266], [0.031], [0.304]])
- [xyz1](#) = np.array([-0.665994, 0.308907, 0.665997])
- [xyz2](#) = np.array([-0.71637, 0.398236, 0.71637])
- [xyz3](#) = np.array([0.0292315, 0.541703, 0.91522])
- [xyz4](#) = np.array([0.158575, 0.957389, 0.50314])
- [xyz5](#) = np.array([-0.35985, 0.225772, 0.924447])
- [xyz6](#) = np.array([-0.666147, 0.308929, 0.666111])
- [frame1](#) = Frame.from_euler_3([xyz1](#), [abc1](#))
- [frame2](#) = Frame.from_euler_3([xyz2](#), [abc2](#))
- [frame3](#) = Frame.from_euler_3([xyz3](#), [abc3](#))
- [frame4](#) = Frame.from_euler_3([xyz4](#), [abc4](#))
- [frame5](#) = Frame.from_euler_3([xyz5](#), [abc5](#))
- [frame6](#) = Frame.from_euler_3([xyz6](#), [abc6](#))
- list [frames](#) = [[frame1](#), [frame2](#), [frame3](#), [frame4](#), [frame5](#), [frame6](#)]
- [trajectory](#) = RobotTrajectory([robot](#), [frames](#))
- [motion](#)

6.1.1 Variable Documentation

6.1.1.1 abc1

```
simulation.abc1 = np.array([[ -0.266], [ 0.031], [ 0.304]])
```

6.1.1.2 abc2

```
simulation.abc2 = np.array([[ -0.232], [ 0.192], [ 0.357]])
```

6.1.1.3 abc3

```
simulation.abc3 = np.array([[ -0.076], [ 0.225], [ 0.461]])
```

6.1.1.4 abc4

```
simulation.abc4 = np.array([[ -0.014], [ 0.262], [ 0.262]])
```

6.1.1.5 abc5

```
simulation.abc5 = np.array([[ -0.221], [ 0.020], [ 0.482]])
```

6.1.1.6 abc6

```
simulation.abc6 = np.array([[ -0.266], [ 0.031], [ 0.304]])
```

6.1.1.7 category

```
simulation.category
```

6.1.1.8 dh_a

```
list simulation.dh_a = [0, 0, 0, 0, 0, 0]
```

6.1.1.9 dh_alpha

```
list simulation.dh_alpha = [-1.5708, 1.5708, 0, -1.5708, 1.5708, 0]
```

6.1.1.10 dh_d

```
list simulation.dh_d = [0.25, 0.15, 0.2, 0, 0, 0.1]
```

6.1.1.11 dh_params

```
simulation.dh_params = np.stack((dh_d,dh_a,dh_alpha,dh_theta), axis=-1)
```

6.1.1.12 dh_theta

```
list simulation.dh_theta = [0, 0, 0, 0, 0, 0]
```

6.1.1.13 frame1

```
simulation.frame1 = Frame.from_euler_3(xyz1, abc1)
```

6.1.1.14 frame2

```
simulation.frame2 = Frame.from_euler_3(xyz2, abc2)
```

6.1.1.15 frame3

```
simulation.frame3 = Frame.from_euler_3(xyz3, abc3)
```

6.1.1.16 frame4

```
simulation.frame4 = Frame.from_euler_3(xyz4, abc4)
```

6.1.1.17 frame5

```
simulation.frame5 = Frame.from_euler_3(xyz5, abc5)
```

6.1.1.18 frame6

```
simulation.frame6 = Frame.from_euler_3(xyz6, abc6)
```

6.1.1.19 frames

```
list simulation.frames = [frame1, frame2, frame3, frame4, frame5, frame6]
```

6.1.1.20 motion

```
simulation.motion
```

6.1.1.21 robot

```
simulation.robot = RobotSerial(dh_params)
```

6.1.1.22 trajectory

```
simulation.trajectory = RobotTrajectory(robot, frames)
```

6.1.1.23 xyz1

```
simulation.xyz1 = np.array([-0.665994, 0.308907, 0.665997])
```

6.1.1.24 xyz2

```
simulation.xyz2 = np.array([-0.71637, 0.398236, 0.71637])
```

6.1.1.25 xyz3

```
simulation.xyz3 = np.array([0.0292315, 0.541703, 0.91522])
```

6.1.1.26 xyz4

```
simulation.xyz4 = np.array([0.158575, 0.957389, 0.50314])
```

6.1.1.27 xyz5

```
simulation.xyz5 = np.array([-0.35985, 0.225772, 0.924447])
```

6.1.1.28 xyz6

```
simulation.xyz6 = np.array([-0.666147, 0.308929, 0.666111])
```


Chapter 7

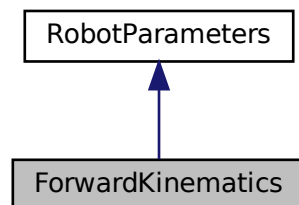
Class Documentation

7.1 ForwardKinematics Class Reference

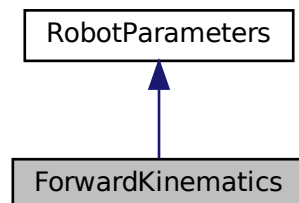
Definition of the Forward Kinematics Class.

```
#include <forward_kinematics.hpp>
```

Inheritance diagram for ForwardKinematics:



Collaboration diagram for ForwardKinematics:



Public Member Functions

- Eigen::Matrix4d [calculate_TF](#) (int i, Eigen::MatrixXd _dh_matrix)
Calculate the DH transformation matrix for each joint pair.
- Eigen::Matrix4d [solve_fk](#) (Eigen::MatrixXd _dh_matrix)
Solve the forward kinematics for manipulator.

Public Attributes

- double [euler_x](#)
- double [euler_y](#)
- double [euler_z](#)

7.1.1 Detailed Description

Definition of the Forward Kinematics Class.

7.1.2 Member Function Documentation

7.1.2.1 [calculate_TF\(\)](#)

```
Matrix4d ForwardKinematics::calculate_TF (
    int i,
    Eigen::MatrixXd _dh_matrix )
```

Calculate the DH transformation matrix for each joint pair.

Parameters

<i>i</i>	integer value denoting the row of _dh_matrix to be considered
----------	---

Returns

Eigen::Matrix<double, 4, 4> Returns the transformation matrix

7.1.2.2 [solve_fk\(\)](#)

```
Matrix4d ForwardKinematics::solve_fk (
    Eigen::MatrixXd _dh_matrix )
```

Solve the forward kinematics for manipulator.

Returns

Eigen::Matrix<double, 4, 4> Returns the final Homogeneous transformation matrix

7.1.3 Member Data Documentation

7.1.3.1 euler_x

```
double ForwardKinematics::euler_x
```

7.1.3.2 euler_y

```
double ForwardKinematics::euler_y
```

7.1.3.3 euler_z

```
double ForwardKinematics::euler_z
```

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

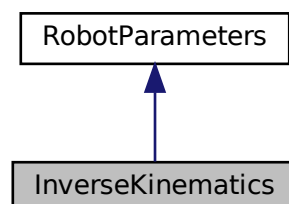
- [include/forward_kinematics.hpp](#)
- [app/forward_kinematics.cpp](#)

7.2 InverseKinematics Class Reference

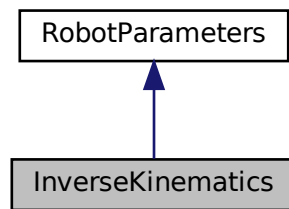
Definition of Inverse Kinematics Class.

```
#include <inverse_kinematics.hpp>
```

Inheritance diagram for InverseKinematics:



Collaboration diagram for InverseKinematics:



Public Member Functions

- [InverseKinematics](#) ()
Construct a new Inverse Kinematics object to assign default values.
- void [set_eff_rotation](#) (Eigen::Matrix3d _R_E)
Sets the end effector rotation matrix.
- Eigen::Matrix3d [get_eff_rotation](#) ()
Gets the end effector rotation matrix.
- Eigen::Vector3d [get_eff_position](#) ()
Gets the end effector position.
- void [set_eff_position](#) (Eigen::Vector3d eff_position)
Sets the end effector position.
- std::vector< double > [solve_ik](#) ()
Method to solve the inverse kinematics of the Manipulator.

Additional Inherited Members

7.2.1 Detailed Description

Definition of Inverse Kinematics Class.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 InverseKinematics()

```
InverseKinematics::InverseKinematics ( )
```

Construct a new Inverse Kinematics object to assign default values.

7.2.3 Member Function Documentation

7.2.3.1 `get_eff_position()`

```
Eigen::Vector3d InverseKinematics::get_eff_position ( )
```

Gets the end effector position.

Returns

`std::vector<double>` Returns a vector containing end effector position

7.2.3.2 `get_eff_rotation()`

```
Eigen::Matrix3d InverseKinematics::get_eff_rotation ( )
```

Gets the end effector rotation matrix.

Returns

`Eigen::Matrix3d` Returns End effector rotation matrix

7.2.3.3 `set_eff_position()`

```
void InverseKinematics::set_eff_position (
    Eigen::Vector3d eff_position )
```

Sets the end effector position.

Parameters

<i>eff_position</i>	Vector containing end effector position
---------------------	---

7.2.3.4 `set_eff_rotation()`

```
void InverseKinematics::set_eff_rotation (
    Eigen::Matrix3d _R_E )
```

Sets the end effector rotation matrix.

Parameters

${}_{-R}^{\leftarrow E}$	Rotation matrix for end effector with respect to base frame
--------------------------	---

7.2.3.5 solve_ik()

```
vector< double > InverseKinematics::solve_ik ( )
```

Method to solve the inverse kinematics of the Manipulator.

Returns

```
std::vector<double>
```

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

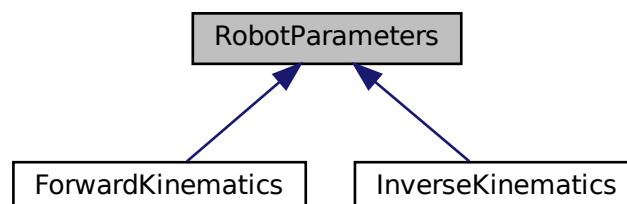
- [include/inverse_kinematics.hpp](#)
- [app/inverse_kinematics.cpp](#)

7.3 RobotParameters Class Reference

Definition of the Robot Parameter Class.

```
#include <robot_parameters.hpp>
```

Inheritance diagram for RobotParameters:



Public Member Functions

- [RobotParameters](#) ()
Construct a new Robot Parameters object to assign default values.
- void [set_dh_parameters](#) ()
Sets the DH parameters.
- Eigen::MatrixXd [get_dh_parameters](#) ()
Compute the dh parameters matrix.
- std::vector< double > [get_robot_angles](#) ()
Gets the robot angles.
- void [set_robot_angles](#) (std::vector< double > robot_angles)
Sets the robot angles.

Public Attributes

- std::string [robot_name](#)

7.3.1 Detailed Description

Definition of the Robot Parameter Class.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 RobotParameters()

```
RobotParameters::RobotParameters ( )
```

Construct a new Robot Parameters object to assign default values.

7.3.3 Member Function Documentation

7.3.3.1 get_dh_parameters()

```
MatrixXd RobotParameters::get_dh_parameters ( )
```

Compute the dh parameters matrix.

Returns

Eigen::MatrixXd Returns DH matrix

7.3.3.2 get_robot_angles()

```
vector< double > RobotParameters::get_robot_angles ( )
```

Gets the robot angles.

Returns

std::vector<double> Returns the robot angles

7.3.3.3 set_dh_parameters()

```
void RobotParameters::set_dh_parameters ( )
```

Sets the DH parameters.

7.3.3.4 set_robot_angles()

```
void RobotParameters::set_robot_angles (
    std::vector< double > robot_angles )
```

Sets the robot angles.

Parameters

<i>robot_angles</i>	Sets the robot angles from the ik solver output
---------------------	---

7.3.4 Member Data Documentation

7.3.4.1 robot_name

```
std::string RobotParameters::robot_name
```

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

- [include/robot_parameters.hpp](#)
- [app/robot_parameters.cpp](#)

Chapter 8

File Documentation

8.1 app/CMakeLists.txt File Reference

Functions

- [add_executable](#) (ik_solver main.cpp robot_parameters.cpp forward_kinematics.cpp inverse_kinematics.cpp) include_directories(\$
- include [find_package](#) (PythonLibs REQUIRED) include_directories(\$

8.1.1 Function Documentation

8.1.1.1 add_executable()

```
add_executable (
    ik_solver main.cpp robot_parameters.cpp forward_kinematics.cpp inverse_kinematics.
    cpp )
```

8.1.1.2 find_package()

```
include find_package (
    PythonLibs REQUIRED )
```

8.2 test/CMakeLists.txt File Reference

Functions

- [set](#) (GTEST_SHUFFLE 1) [add_executable](#)(code_test main.cpp code_test.cpp ../app/robot_parameters.cpp ../app/forward_kinematics.cpp ../app/inverse_kinematics.cpp) target_include_directories(code_test PUBLIC ../vendor/googletest/googletest/include \$

8.2.1 Function Documentation

8.2.1.1 set()

```
set (
    GTEST_SHUFFLE 1 )
```

8.3 app/forward_kinematics.cpp File Reference

Program to define the Methods of Forward Kinematics Class.

```
#include "../include/forward_kinematics.hpp"
#include <iostream>
Include dependency graph for forward_kinematics.cpp:
```

8.3.1 Detailed Description

Program to define the Methods of Forward Kinematics Class.

Author

Driver : Tanmay Haldankar (tanmayh@umd.edu), Navigator: Sanchit Kedia (sanchit@terpmail.umd.edu), Design Keeper: Qamar Syed (qsyed@umd.edu)

Version

0.12

Date

2022-10-18

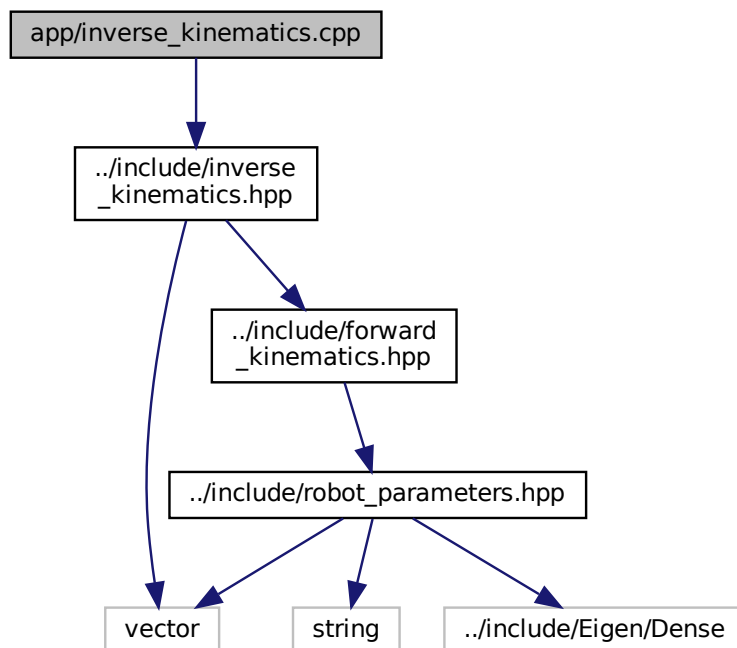
Copyright

MIT License (c)

8.4 app/inverse_kinematics.cpp File Reference

```
#include "../include/inverse_kinematics.hpp"
```

Include dependency graph for inverse_kinematics.cpp:

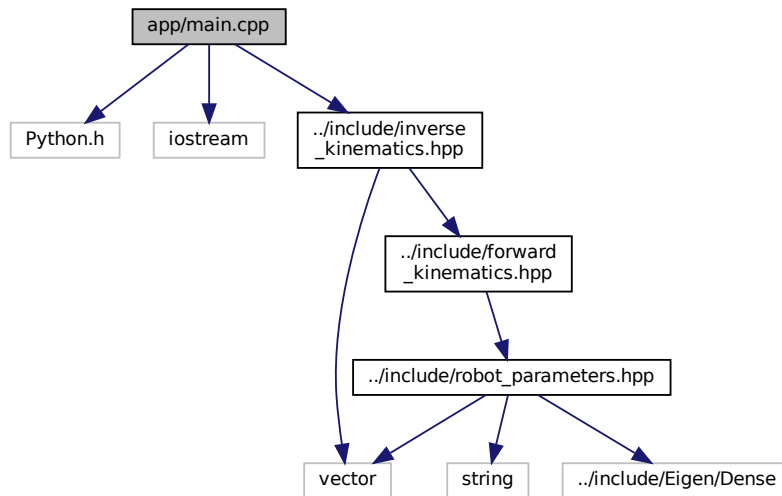


8.5 app/main.cpp File Reference

Program to execute the inverse kinematics and forward kinematics.

```
#include <Python.h>
#include <iostream>
#include "../include/inverse_kinematics.hpp"
```

Include dependency graph for main.cpp:



Functions

- `int main ()`

Main Function that calculates the inverse kinematics solution, verifies it by performing forward kinematics and simulates the robot trajectory.

8.5.1 Detailed Description

Program to execute the inverse kinematics and forward kinematics.

Author

Driver : Sanchit Kedia (sanchit@terpmail.umd.edu), Navigator: Tanmay Haldankar (tanmayh@umd.edu), Design Keeper: Qamar Syed (qsyed@umd.edu)

Version

0.2

Date

2022-10-18

Copyright

MIT License (c)

8.5.2 Function Documentation

8.5.2.1 main()

```
int main ( )
```

Main Function that calculates the inverse kinematics solution, verifies it by performing forward kinematics and simulates the robot trajectory.

Returns

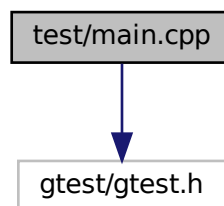
int 0

8.6 test/main.cpp File Reference

Program to use google test for unit testing.

```
#include <gtest/gtest.h>
```

Include dependency graph for main.cpp:



Functions

- int [main](#) (int argc, char **argv)

8.6.1 Detailed Description

Program to use google test for unit testing.

Author

Driver : Sanchit Kedia (sanchit@terpmail.umd.edu), Navigator: Tanmay Haldankar (tanmayh@umd.edu), Design Keeper: Qamar Syed (qsyed@umd.edu)

Version

0.1

Date

2022-10-13

Copyright

MIT License (c)

8.6.2 Function Documentation

8.6.2.1 main()

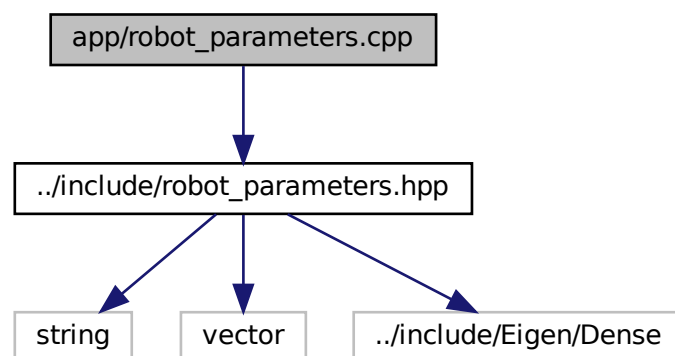
```
int main (
    int argc,
    char ** argv )
```

8.7 app/robot_parameters.cpp File Reference

Program to define the Methods of Robot Parameters Class.

```
#include "../include/robot_parameters.hpp"
```

Include dependency graph for robot_parameters.cpp:



8.7.1 Detailed Description

Program to define the Methods of Robot Parameters Class.

Author

Driver : Sanchit Kedia (sanchit@terpmail.umd.edu), Navigator: Tanmay Haldankar (tanmayh@umd.edu), Design Keeper: Qamar Syed (qsyed@umd.edu)

Version

0.12

Date

2022-10-28

Copyright

MIT License (c)

8.8 app/simulation.py File Reference

Namespaces

- [simulation](#)

Variables

- [simulation.category](#)
- list [simulation.dh_d](#) = [0.25, 0.15, 0.2, 0, 0, 0.1]
- list [simulation.dh_a](#) = [0, 0, 0, 0, 0, 0]
- list [simulation.dh_alpha](#) = [-1.5708, 1.5708, 0, -1.5708, 1.5708, 0]
- list [simulation.dh_theta](#) = [0, 0, 0, 0, 0, 0]
- [simulation.dh_params](#) = np.stack((dh_d,dh_a,dh_alpha,dh_theta), axis=-1)
- [simulation.robot](#) = RobotSerial(dh_params)
- [simulation.abc1](#) = np.array([[[-0.266], [0.031], [0.304]]])
- [simulation.abc2](#) = np.array([[[-0.232], [0.192], [0.357]]])
- [simulation.abc3](#) = np.array([[[-0.076], [0.225], [0.461]]])
- [simulation.abc4](#) = np.array([[[-0.014], [0.262], [0.262]]])
- [simulation.abc5](#) = np.array([[[-0.221], [0.020], [0.482]]])
- [simulation.abc6](#) = np.array([[[-0.266], [0.031], [0.304]]])
- [simulation.xyz1](#) = np.array([-0.665994, 0.308907, 0.665997])
- [simulation.xyz2](#) = np.array([-0.71637, 0.398236, 0.71637])
- [simulation.xyz3](#) = np.array([0.0292315, 0.541703, 0.91522])
- [simulation.xyz4](#) = np.array([0.158575, 0.957389, 0.50314])
- [simulation.xyz5](#) = np.array([-0.35985, 0.225772, 0.924447])
- [simulation.xyz6](#) = np.array([-0.666147, 0.308929, 0.666111])
- [simulation.frame1](#) = Frame.from_euler_3(xyz1, abc1)
- [simulation.frame2](#) = Frame.from_euler_3(xyz2, abc2)
- [simulation.frame3](#) = Frame.from_euler_3(xyz3, abc3)
- [simulation.frame4](#) = Frame.from_euler_3(xyz4, abc4)
- [simulation.frame5](#) = Frame.from_euler_3(xyz5, abc5)
- [simulation.frame6](#) = Frame.from_euler_3(xyz6, abc6)
- list [simulation.frames](#) = [frame1, frame2, frame3, frame4, frame5, frame6]
- [simulation.trajectory](#) = RobotTrajectory(robot, frames)
- [simulation.motion](#)

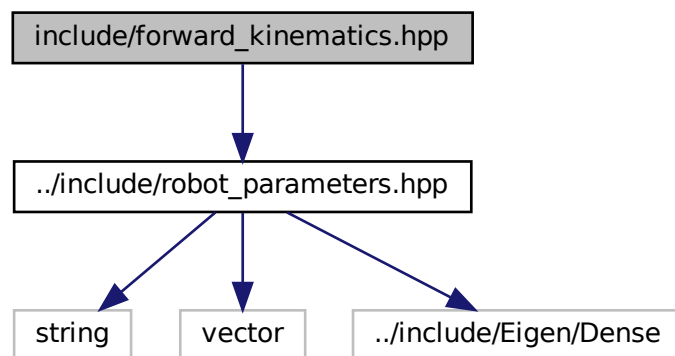
8.9 docs/introduction.txt File Reference

8.10 include/forward_kinematics.hpp File Reference

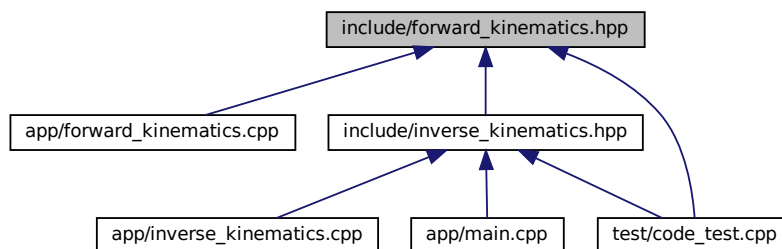
Definition of Forward Kinematics class and Declaration of its Methods.

```
#include "../include/robot_parameters.hpp"
```

Include dependency graph for forward_kinematics.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [ForwardKinematics](#)

Definition of the Forward Kinematics Class.

8.10.1 Detailed Description

Definition of Forward Kinematics class and Declaration of its Methods.

Author

Driver : Tanmay Haldankar (tanmayh@umd.edu), Navigator: Sanchit Kedia (sanchit@terpmail.umd.edu), Design Keeper: Qamar Syed (qsyed@umd.edu)

Version

0.3

Date

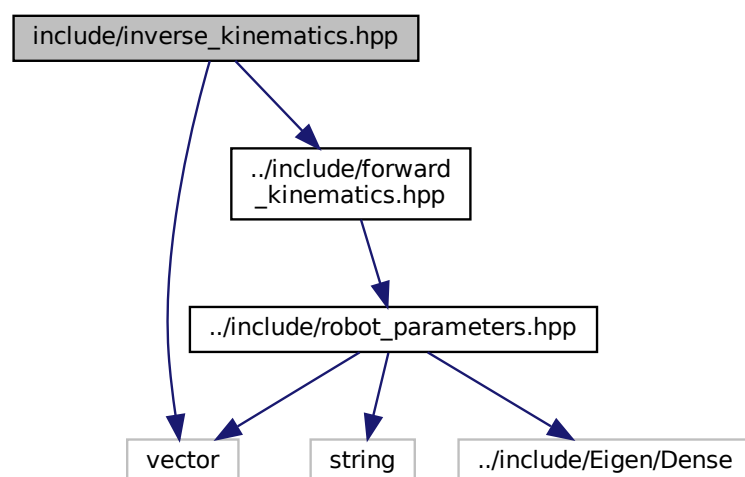
2022-10-28

Copyright

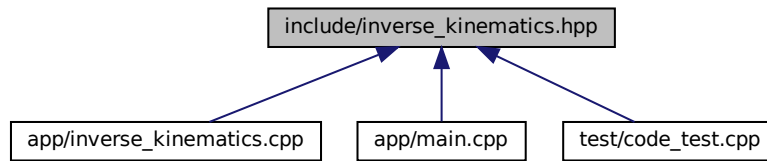
MIT License (c)

8.11 include/inverse_kinematics.hpp File Reference

```
#include <vector>
#include "../include/forward_kinematics.hpp"
Include dependency graph for inverse_kinematics.hpp:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [InverseKinematics](#)

Definition of Inverse Kinematics Class.

8.11.1 Detailed Description

Author

Driver : Sanchit Kedia (sanchit@terpmail.umd.edu), Navigator: Tanmay Haldankar (tanmayh@umd.edu), Design Keeper: Qamar Syed (qsyed@umd.edu)

Version

0.1

Date

2022-10-21

Copyright

MIT License (c)

Author

Driver : Tanmay Haldankar (tanmayh@umd.edu), Navigator: Sanchit Kedia (sanchit@terpmail.umd.edu), Design Keeper: Qamar Syed (qsyed@umd.edu)

Version

0.3

Date

2022-10-28

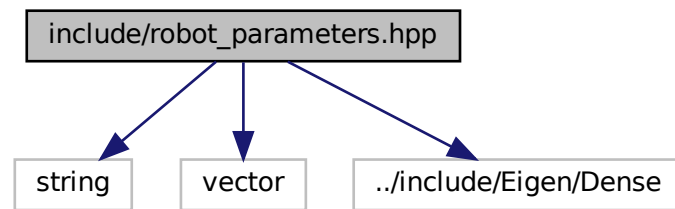
Copyright

MIT License (c)

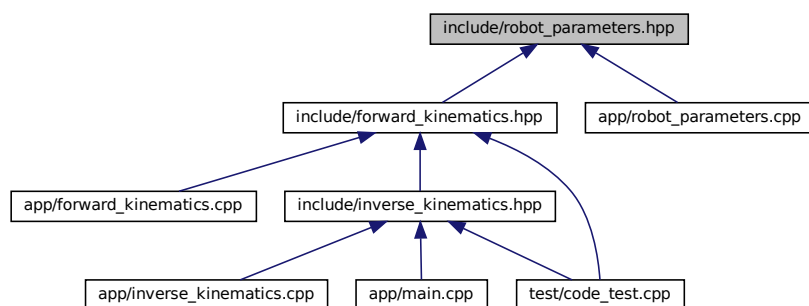
8.12 include/robot_parameters.hpp File Reference

Definition of Robot Parameters class and Declaration of its Methods.

```
#include <string>
#include <vector>
#include "../include/Eigen/Dense"
Include dependency graph for robot_parameters.hpp:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [RobotParameters](#)

Definition of the Robot Parameter Class.

8.12.1 Detailed Description

Definition of Robot Parameters class and Declaration of its Methods.

Author

Driver : Sanchit Kedia (sanchit@terpmail.umd.edu), Navigator: Tanmay Haldankar (tanmayh@umd.edu), Design Keeper: Qamar Syed (qsyed@umd.edu)

Version

0.1

Date

2022-10-15

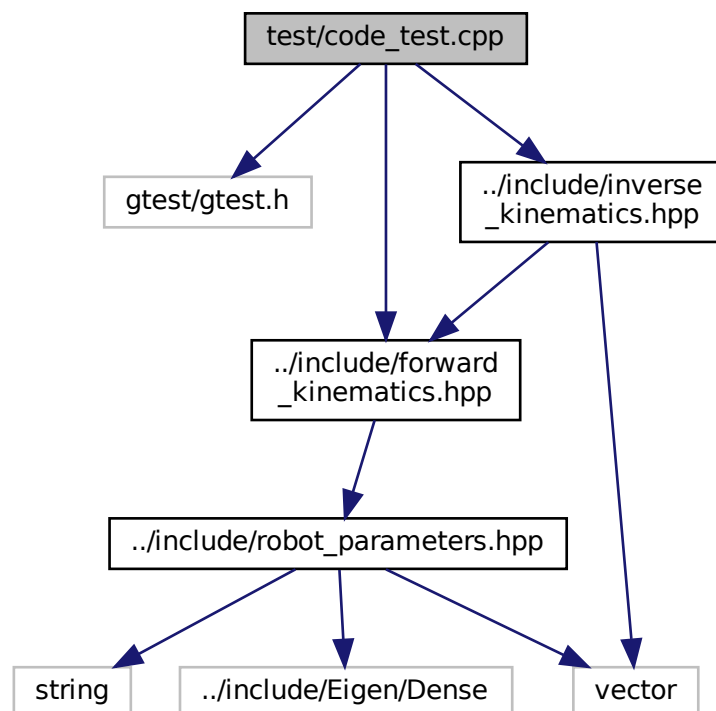
Copyright

MIT License (c)

8.13 test/code_test.cpp File Reference

Program to perform unit testing.

```
#include <gtest/gtest.h>
#include "../include/forward_kinematics.hpp"
#include "../include/inverse_kinematics.hpp"
Include dependency graph for code_test.cpp:
```



Functions

- **TEST** (Robot_Parameters, CheckAngles)
Construct a new TEST to check if the robot angles vector is empty.
- **TEST** (Robot_Parameters, CheckDH)
Construct a new TEST to check the size of the DH Parameters matrix.
- **TEST** (Robot_Parameters, CheckSetAngles)
Construct a new TEST to check if the robot angles are being set correctly.
- **TEST** (Forward_Kinematics, check_calculateTF)
Construct a new TEST to check the size of the DH transformation matrix.
- **TEST** (Forward_Kinematics, check_solvefk)
Construct a new TEST to check the size of final homogeneous transformation matrix.
- **TEST** (Inverse_Kinematics, Check_solveik)
Construct a new TEST to check the output of solve_ik function.
- **TEST** (Inverse_Kinematics, CheckSetIKAngles)
Construct a new TEST to check the size of end effector Rotation Matrix.
- **TEST** (Inverse_Kinematics, CheckSetIKPosition)
Construct a new test to check the size of end effector Position Vector.

8.13.1 Detailed Description

Program to perform unit testing.

Author

Driver : Sanchit Kedia (sanchit@terpmail.umd.edu), Navigator: Tanmay Haldankar (tanmayh@umd.edu), Design Keeper: Qamar Syed (qsyed@umd.edu)

Version

0.3

Date

2022-10-15

Copyright

MIT License (c)

8.13.2 Function Documentation

8.13.2.1 TEST() [1/8]

```
TEST (
    Forward_Kinematics ,
    check_calculateTF )
```

Construct a new TEST to check the size of the DH transformation matrix.

8.13.2.2 TEST() [2/8]

```
TEST (
    Forward_Kinematics ,
    check_solvefk )
```

Construct a new TEST to check the size of final homogeneous transformation matrix.

8.13.2.3 TEST() [3/8]

```
TEST (
    Inverse_Kinematics ,
    Check_solveik )
```

Construct a new TEST to check the output of solve_ik function.

8.13.2.4 TEST() [4/8]

```
TEST (
    Inverse_Kinematics ,
    CheckSetIKAngles )
```

Construct a new TEST to check the size of end effector Rotation Matrix.

8.13.2.5 TEST() [5/8]

```
TEST (
    Inverse_Kinematics ,
    CheckSetIKPosition )
```

Construct a new test to check the size of end effector Position Vector.

8.13.2.6 TEST() [6/8]

```
TEST (
    Robot_Parameters ,
    CheckAngles )
```

Construct a new TEST to check if the robot angles vector is empty.

8.13.2.7 TEST() [7/8]

```
TEST (
    Robot_Parameters ,
    CheckDH )
```

Construct a new TEST to check the size of the DH Parameters matrix.

8.13.2.8 TEST() [8/8]

```
TEST (
    Robot_Parameters ,
    CheckSetAngles )
```

Construct a new TEST to check if the robot angles are being set correctly.

Index

- abc1
 - simulation, [11](#)
- abc2
 - simulation, [12](#)
- abc3
 - simulation, [12](#)
- abc4
 - simulation, [12](#)
- abc5
 - simulation, [12](#)
- abc6
 - simulation, [12](#)
- add_executable
 - CMakeLists.txt, [25](#)
- app/CMakeLists.txt, [25](#)
- app/forward_kinematics.cpp, [26](#)
- app/inverse_kinematics.cpp, [27](#)
- app/main.cpp, [27](#)
- app/robot_parameters.cpp, [30](#)
- app/simulation.py, [31](#)
- calculate_TF
 - ForwardKinematics, [18](#)
- category
 - simulation, [12](#)
- CMakeLists.txt
 - add_executable, [25](#)
 - find_package, [25](#)
 - set, [26](#)
- code_test.cpp
 - TEST, [37–39](#)
- dh_a
 - simulation, [12](#)
- dh_alpha
 - simulation, [12](#)
- dh_d
 - simulation, [13](#)
- dh_params
 - simulation, [13](#)
- dh_theta
 - simulation, [13](#)
- docs/introduction.txt, [32](#)
- euler_x
 - ForwardKinematics, [19](#)
- euler_y
 - ForwardKinematics, [19](#)
- euler_z
 - ForwardKinematics, [19](#)
- find_package
 - CMakeLists.txt, [25](#)
- ForwardKinematics, [17](#)
 - calculate_TF, [18](#)
 - euler_x, [19](#)
 - euler_y, [19](#)
 - euler_z, [19](#)
 - solve_fk, [18](#)
- frame1
 - simulation, [13](#)
- frame2
 - simulation, [13](#)
- frame3
 - simulation, [13](#)
- frame4
 - simulation, [13](#)
- frame5
 - simulation, [13](#)
- frame6
 - simulation, [14](#)
- frames
 - simulation, [14](#)
- get_dh_parameters
 - RobotParameters, [23](#)
- get_eff_position
 - InverseKinematics, [21](#)
- get_eff_rotation
 - InverseKinematics, [21](#)
- get_robot_angles
 - RobotParameters, [23](#)
- include/forward_kinematics.hpp, [32](#)
- include/inverse_kinematics.hpp, [33](#)
- include/robot_parameters.hpp, [35](#)
- InverseKinematics, [19](#)
 - get_eff_position, [21](#)
 - get_eff_rotation, [21](#)
 - InverseKinematics, [20](#)
 - set_eff_position, [21](#)
 - set_eff_rotation, [21](#)
 - solve_ik, [22](#)
- main
 - main.cpp, [29, 30](#)
- main.cpp
 - main, [29, 30](#)
- motion
 - simulation, [14](#)
- robot

- simulation, 14
- robot_name
 - RobotParameters, 24
- RobotParameters, 22
 - get_dh_parameters, 23
 - get_robot_angles, 23
 - robot_name, 24
 - RobotParameters, 23
 - set_dh_parameters, 24
 - set_robot_angles, 24
- set
 - CMakeLists.txt, 26
- set_dh_parameters
 - RobotParameters, 24
- set_eff_position
 - InverseKinematics, 21
- set_eff_rotation
 - InverseKinematics, 21
- set_robot_angles
 - RobotParameters, 24
- simulation, 11
 - abc1, 11
 - abc2, 12
 - abc3, 12
 - abc4, 12
 - abc5, 12
 - abc6, 12
 - category, 12
 - dh_a, 12
 - dh_alpha, 12
 - dh_d, 13
 - dh_params, 13
 - dh_theta, 13
 - frame1, 13
 - frame2, 13
 - frame3, 13
 - frame4, 13
 - frame5, 13
 - frame6, 14
 - frames, 14
 - motion, 14
 - robot, 14
 - trajectory, 14
 - xyz1, 14
 - xyz2, 14
 - xyz3, 14
 - xyz4, 15
 - xyz5, 15
 - xyz6, 15
- solve_fk
 - ForwardKinematics, 18
- solve_ik
 - InverseKinematics, 22
- TEST
 - code_test.cpp, 37–39
- test/CMakeLists.txt, 25
- test/code_test.cpp, 36
- test/main.cpp, 29
- trajectory
 - simulation, 14
- xyz1
 - simulation, 14
- xyz2
 - simulation, 14
- xyz3
 - simulation, 14
- xyz4
 - simulation, 15
- xyz5
 - simulation, 15
- xyz6
 - simulation, 15