WBM::ClassDiagram

WBMBase

+fixed_link: string
+init_wf_R_b: matrix
+init_wf_p_b: vector
+g_wf: vector
+ndof: integer
+frict_coeff: struct
+joint_limits: struct [readOnly]
+robot_model: wbmBaseRobotModel [readOnly]
#DF_ROBOT_MODEL: string [readOnly]
#MAX_NUM_JOINTS: integer [readOnly]

+WBMBase(out obj: WBMBase, robot_model: wbmBaseRobotModel)
+copy(out newObj: WBMBase, obj)
+delete()
+initModel(obj)
+initModel(obj, urdf_model_name: string)
+initModelURDF(obj, urdf_file_name: string)
+setWorldFrame(obj, wf_R_b: matrix, wf_p_b: vector)
+setWorldFrame(obj, wf_R_b: matrix, wf_p_b: vector, g_wf: vector)
+setInitWorldFrame(obj, wf_R_b: matrix, wf_p_b: vector)
+setInitWorldFrame(obj, wf_R_b: matrix, wf_p_b: vector, g_wf: vector)
+getWorldFrameFromFixLnk(out wf_p_b: vector, out wf_R_b: matrix, obj, urdf_fixed_link: string)
+getWorldFrameFromFixLnk(out wf_p_b: vector, out wf_R_b: matrix, obj, urdf_fixed_link: string, q_j: vector)
+getWorldFrameFromDfltFixLnk(out wf_p_b: vector, out wf_R_b: matrix, obj)
+getWorldFrameFromDfltFixLnk(out wf_p_b: vector, out wf_R_b: matrix, obj, q_j: vector)
+setState(q_j: vector, dq_j: vector, v_b: vector)
+getState(out vqT_b: vector, out q_j: vector, out v_b: vector, out dq_j: vector)
+getFloatingBaseState(out stFltb: wbmFltgBaseState)
+transformationMatrix(out wf_H_lnk: matrix, obj)
+transformationMatrix(out wf_H_lnk: matrix, obj, urdf_link_name: string)
+transformationMatrix(out wf_H_lnk: matrix, obj, wf_R_b: matrix, wf_p_b: vector, q_j: vector)
+transformationMatrix(out wf_H_lnk: matrix, obj, wf_R_b: matrix, wf_p_b: vector, q_j: vector, urdf_link_name: string)
+massMatrix(out M: matrix, obj)
+massMatrix(out M: matrix, obj, wf_R_b: matrix, wf_p_b: vector, q_j: vector)
+getJointLimits(out jmin: vector, out jmax: vector, obj)
+hJointLimits(out resv: vector, obj, q_j: vector)
+generalizedBiasAcc(out dv_b: vector, obj, M: matrix, c_qv: vector, ddq_j: vector)
+inverseDynamics(out tau: vector, obj)
+inverseDynamics(out tau_j: vector, obj, dq_j: vector, ddq_j: vector)
+inverseDynamics(out tau_j: vector, obj, wf_R_b: matrix, wf_p_b: vector, q_j: vector, dq_j: vector, dv_b: vector)
+inverseDynamics(out tau_j: vector, obj, wf_R_b: matrix, wf_p_b: vector, q_j: vector, dq_j: vector, ddq_j: vector, dv_b: vector)
+jacobian(out wf_J_lnk: matrix, obj)
+jacobian(out wf_J_lnk: matrix, obj, urdf_link_name: string)
+jacobian(out wf_J_lnk: matrix, obj, wf_R_b: matrix, wf_p_b: vector, q_j: vector)
+jacobian(out wf_J_lnk: matrix, obj, wf_R_b: matrix, wf_p_b: vector, q_j: vector, urdf_link_name: string)
+dJdq(out dJdq_lnk: vector, obj)
+dJdq(out dJdq_lnk: vector, obj, urdf_link_name: string)
+dJdq(out dJdq_lnk: vector, obj, wf_R_b: matrix, wf_p_b: vector, q_j: vector, dq_j: vector, v_b: vector)
+dJdq(out dJdq_lnk: vector, obj, wf_R_b: matrix, wf_p_b: vector, q_j: vector, dq_j: vector, v_b: vector, urdf_link_name: string)
+centroidalMomentum(out h_c: struct)
+centroidalMomentum(out h_c: vector, obj, wf_R_b: matrix, wf_p_b: vector, q_j: vector, dq_j: vector, v_b: vector)
+forwardKinematics(out vqT_lnk: vector, obj)
+forwardKinematics(out vqT_lnk: vector, obj, urdf_link_name: string)
+forwardKinematics(out vqT_lnk: vector, obj, wf_R_b: matrix, wf_p_b: vector, q_j: vector)
+forwardKinematics(out vqT_lnk: vector, obj, wf_R_b: matrix, wf_p_b: vector, q_j: vector, urdf_link_name: string)
+generalizedBiasForces(out c_qv: vector)
+generalizedBiasForces(out c_qv: vector, obj, wf_R_b: matrix, wf_p_b: vector, q_j: vector, dq_j: vector, v_b: vector)
+generalizedForces(out tau_gen: vector, obj, wf_R_b: matrix, wf_p_b: vector, q_j: vector, dq_j: vector, v_b: vector, 2_e: vector)
+generalizedForces(out tau_gen: vector, wf_R_b: matrix, wf_p_b: vector, q_j: vector, dq_j: vector, v_b: vector, 2_e: vector, f_e: vector)
+coriolisBiasForces(out c_qv: vector)
+coriolisBiasForces(out c_qv: vector, obj, wf_R_b: matrix, wf_p_b: vector, q_j: vector, dq_j: vector, v_b: vector)
+gravityBiasForces(out g_q: vector)
+gravityBiasForces(out g_q: vector, obj, wf_R_b: matrix, wf_p_b: vector, q_j: vector)
+frictionForces(out tau_fr: vector, obj)
+frictionForces(out tau_fr: vector, obj, dq_j: vector)
+wholeBodyDynamics(out M: matrix, out c_qv: vector, out h_c: vector, obj)
+wholeBodyDynamics(out M: matrix, out c_qv: vector, out h_c: vector, obj, wf_R_b: matrix, wf_p_b: vector, q_j: vector, dq_j: vector, v_b: vector)
+set.fixed_link(obj, rlnk_name: string)
+get.fixed_link(out rlnk_name: string, obj)
+set.init_wf_R_b(obj, wf_R_b: matrix)
+set.init_wf_p_b(out wf_R_b: matrix, obj)
+set.init_wf_p_b(obj, wf_p_b: vector)
+get.init_wf_p_b(out wf_p_b: vector, obj)
+set.g_wf(obj, g_wf: vector)
+get.g_wf(out g_wf: vector, obj)
+set.ndof(obj, ndof: integer)
+get.ndof(out ndof: integer, obj)
+set.frict_coeff(obj, frict_coeff: struct)
+get.frict_coeff(out frict_coeff: struct, obj)
+get.joint_limits(out jlimits: struct, obj)
+get.robot_model(out robot_model: wbmBaseRobotModel, obj)
-initBaseModel(obj, prec: integer)
-initWBM(obj, robot_model: wbmBaseRobotModel)
-computeNewWorld2Base(obj, urdf_link_name: string): vector
-computeNewWorld2Base(out vw_p_b: vector, out nw_R_b: matrix, obj, urdf_link_name: string, q_j: vector)

«dataType» wbmBaseRobotModel

+ndof: integer
+urdf_robot_name: string
+urdf_fixed_link: string
+wf_R_b_init: vector
+wf_p_b_init: vector
+g_wf: vector
+frict_coeff.v: vector
+frict_coeff.c: vector
+jlmts.lwr: vector
+jlmts.upr: vector

Base model parameters of a floating base robot.

«dataType» wbmBaseRobotParams

+nsf2fixLnk: logical

Base model and configuration parameters of a floating base robot. This data type is useful e.g. for interfaces.

Since every robot is different, you can specify the configuration parameters of your robot in your own data type.

Robot.YourRobotName_ModelType
«dataType» yourSpecificRobotConfig

«dataType» wbmHumanoidConfig

+pos_head: vector
+pos_torso: vector
+pos_left_arm: vector
+pos_left_leg: vector
+pos_right_arm: vector
+pos_right_leg: vector
+pos_left_hand: vector
+pos_right_hand: vector
+pos_left_foot: vector
+pos_right_foot: vector

General configuration structure of a humanoid robot.

«dataType» wbmPayloadLink

+urdf_link_name: string
+lnk_p_cm: vector
+m_rb: double
+I_cm: matrix

Payload at a specified link, e.g. hands with a point mass of a rigid body at a specified position in the frame of the reference (end-effector) link.

«dataType» wbmToolLink

+urdf_link_name: string
+ee_vqT_tt: vector

Tool at a specified end-effector link, i.e. hand/finger, with an orientation and translation relative to the link frame.

«dataType» wbmBody

+chains.name: string[1..*] [readOnly]
+chains.start_idx: integer[1..*] [readOnly]
+chains.end_idx: integer[1..*] [readOnly]
+nChains: integer [readOnly]
+joints.name: string[1..*] [readOnly]
+joints.idx: integer[1..*] [readOnly]
+nJoints: integer [readOnly]

+wbmBody(out obj: wbmbody, chain_names: string[1..*], chain_idx: matrix, joint_names: string[1..*], joint_idx: vector)
+getChainIndices(out jnt_idx: vector, obj, chain_name: string)
+getJointIndex(out jnt_idx: integer, obj, joint_name: string)
+getJointNames(out jnt_names: string[1..*], obj, joint_idx: vector)
+getChainTable(out chn_tbl: table, obj)
+getJointTable(out jnt_tbl: table, obj)

Configuration data to define the body components of the robot.

«dataType» wbmFltgBaseState

+wf_R_b: matrix
+wf_p_b: vector
+v_b: vector

State parameters of the robot floating base. This data type is useful e.g. for interfaces.

«dataType» wbmStateParams

+x_b: matrix
+qt_b: matrix
+q_j: matrix
+dx_b: matrix
+omega_b: matrix
+dq_j: matrix

State parameters of the current state of the robot.

«dataType» wbmBaseRobotConfig

+nCstrs: integer
+ccstr_link_names: cell
+nPlds: integer
+nTools: integer
+stvLen: integer

Initial condition of a floating base robot.

WBM

+stvLen: integer [readOnly]
+vqT_base: vector [readOnly]
+init_vqT_base: vector [readOnly]
+init_stvChi: vector [readOnly]
+init_state: wbmStateParams
+robot_body: wbmBody [readOnly]
+robot_config: wbmBaseRobotConfig [readOnly]
+robot_params: wbmBaseRobotParams [readOnly]
#DF_STIFFNESS: double [readOnly]
#MAX_NUM_TOOLS: integer [readOnly]
#ZERO_CVEC_12: vector [readOnly]
#ZERO_CVEC_6: vector [readOnly]
#nwf2fixLnk: logical

+WBM(out obj: WBM, robot_model: wbmBaseRobotModel, robot_config: wbmBaseRobotConfig)
+WBM(out obj: WBM, robot_model: wbmBaseRobotModel, robot_config: wbmBaseRobotConfig, wf2fixLnk: logical)
+copy(out newObj: WBM, obj)
+delete(obj)
+setWorldFrameAtFixLnk(obj, urdf_fixed_link: string)
+setWorldFrameAtFixLnk(obj, urdf_fixed_link: string, q_j: vector, dq_j: vector, v_b: vector)
+setWorldFrameAtFixLnk(obj, urdf_fixed_link: string, q_j: vector, dq_j: vector, v_b: vector, g_wf: vector)
+updateWorldFrameFromFixLnk(obj)
+updateWorldFrameFromFixLnk(obj, q_j: vector)
+updateWorldFrameFromFixLnk(obj, q_j: vector, dq_j: vector, v_b: vector, g_wf: vector)
+updateInitVQTransformation(obj)
+fkinVQTransformation(out vqT_lnk: vector, obj, urdf_link_name: string, q_j: vector, vqT_b: vector, g_wf: vector)
+contactJacobians(out Jc: matrix, out djcdq: vector, obj, idx_list: vector)
+contactJacobians(out Jc: matrix, out djcdq: vector, obj, wf_R_b: vector, wf_p_b: vector, q_j: vector, dq_j: vector, v_b: vector)
+contactJacobians(out Jc: matrix, out djcdq: vector, obj, wf_R_b: arr: vector, wf_p_b: vector, q_j: vector, dq_j: vector, v_b: vector, idx_list: vector)
+contactForces(out f_c: vector, out tau_gen: struct, obj, tau: vector, Jc: matrix, djcdq: vector, M: matrix, c_qv: vector)
... (numerous method signatures continue)
+wholeBodyDyn(out M: matrix, out c_qv: vector)
+wholeBodyDyn(out M: matrix, out c_qv: vector, out wf_R_b_arr: vector, wf_p_b: vector, q_j: vector, dq_j: vector, v_b: vector)

«interface» handle

+delete()

«dataType» wbmRobotDrawProp

+joints.marker: char
+joints.marker_sz: double
+joints.color
+links.line_width: double
+links.color
+com.marker: char
+com.marker_sz: double
+com.color
+shape.line_width: double
+shape.edge_color
+shape.face_color
+shape.face_alpha: double

Draw properties for the body components of the simulated robot.

«utility» wbmColor

«dataType» wbmSimBody

+shape_geom: struct
+shape_size_sf: matrix
+shape_faces: matrix
+foot_geom: struct
+foot_joints: vector
+foot_base_sz
+foot_base_ds: matrix
+joint_link_names: string[1..*] [readOnly]
+joint_pair_idx: matrix [readOnly]
+nJoints: integer [readOnly]
+nLinks: integer [readOnly]
+nFeets: integer [readOnly]
-mshape_geom.size_sf: matrix
-mshape_geom.faces: matrix
-mfoot_geom.base.sz: vector
-mfoot_geom.base_sz.width: double
-mfoot_geom.base_sz.height: double
-mfoot_geom.shape_ds: double

+wbmSimBody(out obj: wbmSimBody, joint_link_names: string[1..*], joint_pair_idx: matrix, draw_prop: wbmDrawProp)
+set.shape_geom(obj, shape_geom: struct)
+get.shape_geom(out shape_geom: struct, obj)
+set.shape_size_sf(obj, shape_size_sf: matrix)
+set.shape_size_sf(out shape_size_sf: matrix, obj)
+set.shape_faces(obj, shape_faces: matrix)
+set.shape_faces(out shape_faces: matrix, obj)
+set.foot_geom(obj, foot_geom: struct)
+get.foot_geom(out foot_geom: struct, obj)
+set.foot_joints(obj, foot_joints: vector)
+get.foot_joints(out foot_jnts: vector, obj)
+set.foot_base_sz(obj, base_sz: struct)
+get.foot_base_sz(obj, base_sz: struct)
+set.foot_base_sz(obj, foot_base_sz: struct, obj)
+set.foot_shape_ds(obj, foot_ds: matrix)
+get.foot_shape_ds(out foot_shape_ds: matrix, obj)

«utility» wbmErrorMsg

#DIM_MISMATCH: string [readOnly]
#EMPTY_ARRAY: string [readOnly]
#EMPTY_DATA_TYPE: string [readOnly]
#EMPTY_STRING: string [readOnly]
#EMPTY_VECTOR: string [readOnly]
#FILE_NOT_EXIST: string [readOnly]
#IDX_OUT_OF_BOUNDS: string [readOnly]
#LIST_NOT_SORTED: string [readOnly]
#LNK_NOT_IN_LIST: string [readOnly]
#MAX_JOINT_LIMIT: string [readOnly]
#MAX_NUM_LIMIT: string [readOnly]
#MIN_JOINT_LIMIT: string [readOnly]
#NAME_NOT_EXIST: string [readOnly]
#NOT_NUMOG_MAT: string [readOnly]
#NO_LNK_IN_CTC: string [readOnly]
#SINGULAR_MAT: string [readOnly]
#STRING_MISMATCH: string [readOnly]
#UNKNOWN_AXIS_SEQ: string [readOnly]
#UNKNOWN_CTC_MODEL: [readOnly]
#UNKNOWN_EXC: string [readOnly]
#UNKNOWN_MAT_TYPE: string [readOnly]
#UNKNOWN_LNK_NAME: string [readOnly]
#VALUE_IS_INIT: string [readOnly]
#VALUE_LT_ZERO: string [readOnly]
#VALUE_LT_ZERO: string [readOnly]
#VAL_OUT_OF_BOUNDS: string [readOnly]
#WRONG_ARR_SIZE: string [readOnly]
#WRONG_DATA_TYPE: string [readOnly]
#WRONG_MAT_DIM: string [readOnly]
#WRONG_NARGIN: string [readOnly]
#WRONG_VEC_DIM: string [readOnly]
#WRONG_VEC_SIZE: string [readOnly]

«dataType» wbmSimEnvironment

+background_color_opt: string
+ground_shape: matrix
+ground_color
+ground_edge_color
+origin_pt_color
+origin_pt_size: double

Environment settings for the simulation.

«dataType» absSimConfig

#DF_MAIN_POS: vector [readOnly]
#DF_AXES_POS: matrix [readOnly]
#DF_AXES_COLORS: matrix [readOnly]
#DF_AXIS_LIMITS: vector [readOnly]
#DF_GROUND_SHAPE: matrix [readOnly]
#DF_GROUND_COLOR: vector [readOnly]
+hMainFigure: matlab.ui.Figure
+main_title: string
+main_pos: vector
+hAxes: vector
+axes_pos: matrix
+axes_colors: matrix
+axis_limits: vector
+plot_objs: cell

Robot.YourRobotName_ModelType
«dataType» yourOwnSimConfig

General configuration data for the environment and the robot in the simulation.

«dataType» genericSimConfig

#DF_GROUND_COLOR_Z: vector [readOnly]

+genericSimConfig(out obj: genericSimConfig, main_title: string, robot_sim: wbmSimBody, env_settings: wbmSimEnvironment)

Relationship annotations:
- Base model parameters of a floating base robot. (#mwbm_model)
- +model / +config
- +draw_prop
- +robot_body
- Geometry data to define the structure and the shape of the simulated robot.
- +environment
- Environment settings for the simulation.
- +payload_links
- +tool_links
- +body
- +init_state_params
- #mwbm_config
- +ref2fixLnk: logical