WBM::ClassDiagram

WBMBase

+fixed_link: string
+wf_R_b: matrix
+wf_p_b: vector
+g_wf: vector
+ndof: integer
+frict_coeff: struct
+joint_limits: struct {readOnly}
+robot_model: wbmBaseRobotModel {readOnly}
+DFLT_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)
+setInitWorldFrame(obj, wf_R_b: matrix, wf_p_b: vector)
+setInitWorldFrame(obj, wf_R_b: matrix, wf_p_b: vector, g_wf: vector)
+getWorldFrameFromFixedLink(out wf_p_b: vector, out wf_R_b: matrix, obj, urdf_fixed_link: string)
+getWorldFrameFromFixedLink(out wf_p_b: vector, out wf_R_b: matrix, obj, urdf_fixed_link: string, q_j: vector)
+getWorldFrameFromDfltFixedLink(out wf_p_b: vector, out wf_R_b: matrix, obj)
+getWorldFrameFromDfltFixedLink(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)
+massMatrix(out M: matrix, wf_R_b: matrix, wf_p_b: vector, q_j: vector)
+getJointLimits(out jl_lower: vector, out jl_upper: vector)
+isJointLimit(out resv: vector, obj, q_j: vector)
+inverseDynamics(out tau_j: vector, ob, dq_j: vector, ddq_j: vector)
+inverseDynamics(out tau_j: vector, obj, dq_j: vector, ddq_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, v_b: vector, ddq_j: vector)
+inverseDynamics(out tau_j: vector, obj, wf_R_b: matrix, wf_p_b: vector, q_j: vector, dq_j: vector, v_b: vector, ddq_j: vector, dv_b: vector)
+jacobian(out J: matrix, obj)
+jacobian(out J: matrix, obj, urdf_link_name: string)
+jacobian(out J: matrix, obj, wf_R_b: matrix, wf_p_b: vector, q_j: vector)
+jacobian(out J: matrix, obj, wf_R_b: matrix, wf_p_b: vector, q_j: vector, urdf_link_name: string)
+dJdq(out dJ: vector, obj)
+dJdq(out dJ: vector, obj, urdf_link_name: string)
+dJdq(out dJ: vector, obj, wf_R_b: matrix, wf_p_b: vector, q_j: vector, dq_j: vector, v_b: vector)
+dJdq(out dJ: 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: vector)
+centroidalMomentum(out h_c: vector, wf_R_b: matrix, wf_p_b: vector, q_j: vector, dq_j: vector, v_b: vector)
+forwardKinematics(out wf_vqT_lnk: vector, obj)
+forwardKinematics(out wf_vqT_lnk: vector, obj, urdf_link_name: string)
+forwardKinematics(out wf_vqT_lnk: vector, obj, wf_R_b: matrix, wf_p_b: vector, q_j: vector)
+forwardKinematics(out wf_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, wf_R_b: matrix, wf_p_b: vector, q_j: vector, dq_j: vector, v_b: vector)
+generalizedForces(out tau: vector, f_c: vector, Jc_t: matrix)
+generalizedForces(out tau: vector, wf_R_b: matrix, wf_p_b: vector, q_j: vector, dq_j: vector, v_b: vector, f_c: vector, Jc_t: matrix)
+coriolisCentrifugalForces(out tau_c: vector)
+coriolisCentrifugalForces(out tau_c: vector, wf_R_b: matrix, wf_p_b: vector, q_j: vector, dq_j: vector, v_b: vector)
+gravityForces(out tau_g: vector)
+gravityForces(out tau_g: vector, 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, lnk_name: string)
+get.fixed_link(out lnk_name: string, obj)
+set.wf_R_b(obj, new_rotm: matrix)
+get.wf_R_b(out wf_R_b: matrix, obj)
+set.wf_p_b(obj, new_pos: vector)
+get.wf_p_b(out wf_p_b: vector, obj)
+set.g_wf(obj, g: 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 jl: struct, obj)
+get.robot_model(out robot_model: wbmBaseRobotModel, obj)
+dispModel(obj, prec: integer)
-initWBM(obj, robot_model: wbmBaseRobotModel)
-computeNewWorld2Base(obj, urdf_link_name: string): vector
-computeNewWorld2Base(out nw_p_b: vector, out nw_R_b: matrix, obj, urdf_link_name: string, q_j: 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

«use»

«utility»
wbmColor

1
+draw_prop
Draw properties for the body components of the simulated robot.

«dataType»
wbmSimBody
+shape_geom: struct
+shape_size_sf: matrix
+shape_faces: matrix
+foot_geom: struct
+foot_joints: vector
+foot_base_sz
+foot_shape_ds: matrix
+joint_lnk_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.joints: vector
-mfoot_geom.base_sz.width: double
-mfoot_geom.base_sz.height: double
-mfoot_geom.shape_ds: double
+wbmSimBody(out obj: wbmSimBody, joint_lnk_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, size_sf: matrix)
+get.shape_size_sf(out shape_size_sf: matrix, obj)
+set.shape_faces(obj, shape_faces: matrix)
+get.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_jnts: vector)
+get.foot_joints(out foot_jnts: vector, obj)
+set.foot_base_sz(obj, base_sz: struct)
+set.foot_base_sz(obj, base_sz: vector)
+get.foot_base_sz(out foot_base_sz: struct, obj)
+set.foot_shape_ds(obj, foot_ds: matrix)
+get.foot_shape_ds(out foot_shape_ds: matrix, obj)

«use»

«dataType»
wbmRobotBody
+robot_body

Geometry data to define the structure and the shape of the simulated robot.

«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.
+environment

«use»

«dataType»
wbmBaseRobotModel
+ndof: integer
+urdf_robot_name: string
+urdf_fixed_link: string
+wf_R_b: matrix
+wf_p_b: vector
+g_wf: vector
+frict_coeff.v: vector
+frict_coeff.c: vector
+jlim.lwr: vector
+jlim.upr: vector

Base model parameters of a floating base robot.
1
#mwbm_model
1
+model

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

«dataType»
wbmBaseRobotParams
+wf2fixLnk: logical

«use»

WBM
+stvChiInit: vector {readOnly}
+stvLen: integer {readOnly}
+vqTInit: vector {readOnly}
+stvqT: vector {readOnly}
+robot_body: wbmBody {readOnly}
+robot_config: wbmBaseRobotConfig {readOnly}
+robot_params: wbmBaseRobotParams {readOnly}
+init_state: wbmStateParams
+DFLT_STIFFNESS: double {readOnly}
+MAX_NUM_TOOLS: integer {readOnly}
#mwf2fixLnk: 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)
+setWorldFrameAtFixedLink(obj, urdf_fixed_link: string)
+setWorldFrameAtFixedLink(obj, urdf_fixed_link: string, q_j: vector, dq_j: vector, v_b: vector)
+setWorldFrameAtFixedLink(obj, urdf_fixed_link: string, q_j: vector, dq_j: vector, v_b: vector, g_wf: vector)
+updateWorldFrameAtDfltFixedLink(obj)
+updateWorldFrameAtDfltFixedLink(obj, q_j: vector, dq_j: vector, v_b: vector)
+updateWorldFrameAtDfltFixedLink(obj, q_j: vector, dq_j: vector, v_b: vector, g_wf: vector)
+updateInitVQTransformation(obj)
+fkinVQTransformation(out wf_vqT_lnk: vector, obj, urdf_link_name: string, q_j: vector, vqT: vector, g_wf: vector)
+jointAccelerations(out ddq_j: vector, out acc_data: struct, obj, dq_j: vector, tau: vector)
+jointAccelerations(out ddq_j: vector, out acc_data: struct, obj, wf_R_b: matrix, wf_p_b: vector, q_j: vector, dq_j: vector, v_b, vector, tau: vector)
+jointAccelerationsExt(out ddq_j: vector, out acc_data: struct, obj, dq_j: vector, v_b: vector, tau: vector, foot_conf: struct)
+jointAccelerationsExt(out ddq_j: vector, out acc_data: struct, obj, M: matrix, C_qv: vector, dq_j: vector, nu: vector, tau: vector, Jc: matrix, dJcdq: vector, foot_conf: struct)
+jointAccelerationsExt(out ddq_j: vector, out acc_data: struct, obj, wf_R_b: matrix, wf_p_b: vector, q_j: vector, dq_j: vector, v_b: vector, tau: vector, foot_conf: struct)
+forwardDynamics(out dstvChi: vector, obj, t: double, stvChi: vector, fhTrqControl: function_handle)
+forwardDynamicsExt(out dstvChi: vector, obj, t: integer, stvChi: vector, fhTrqControl: function_handle, foot_conf: struct)
+intForwardDynamics(out t: vector, out stmChi: matrix, obj, fhTrqControl: function_handle, tspan: vector, stvChi_0: vector, ode_opt: struct)
+intForwardDynamics(out t: vector, out stmChi: matrix, obj, fhTrqControl: function_handle, tspan: vector, stvChi_0: vector, ode_opt: struct, foot_conf: struct)
+initFootBalanceConfig(out foot_conf: struct, obj, qj_init: vector, feet_on_ground: vector)
+initFootBalanceConfig(out foot_conf: struct, obj, qj_init: vector, feet_on_ground: vector, k_p: double)
+initFootBalanceConfig(out foot_conf: struct, obj, qj_init: vector, feet_on_ground: vector, k_p: double, k_v: double)
+initFootBalanceConfig(out foot_conf: struct, obj, qj_init: vector, veT_lfoot: vector, veT_rfoot: vector, feet_on_ground: vector, k_p: double, k_v: double)
+getFwdDynVisualizationData(out vis_data: struct, obj, stmChi: matrix, fhTrqControl: function_handle, foot_conf: struct)
+setupSimulation(out sim_config: absSimConfig, sim_config: absSimConfig)
+visualizeForwardDynamics(obj, pos_out: matrix, sim_config: absSimConfig, sim_tstep: double, vis_ctrl: struct)
+simulateForwardDynamics(obj, pos_out: matrix, sim_config: absSimConfig, sim_tstep: double, nRpts: integer, vis_ctrl: struct)
+plotCoMTrajectory(obj, stmChi: matrix, prop: struct)
+setLinkPayloads(obj, link_names: string[1..*], pl_data: matrix)
+getPayloadLinks(out pl_links: wbmPayloadLink[0..*], out nPlds: integer, obj)
+getPayloadTable(out pl_tbl: table, obj)
+payloadFrame(out wf_H_pl: matrix, obj, wf_R_b: matrix, wf_p_b: vector, q_j: vector, pl_idx: integer)
+payloadFrame(out wf_H_pl: matrix, obj, wf_R_b: matrix, wf_p_b: vector, q_j: vector)
+payloadFrame(out wf_H_pl: matrix, obj, pl_idx: integer)
+payloadFrame(out wf_H_pl: matrix, obj)
+setToolLinks(obj, ee_link_names: string[1..*], frames_tt: matrix)
+getToolLinks(out tool_links: wbmToolLink[0..*], out nTools: integer, obj)
+getToolTable(out tool_tbl: table, obj)
+updateToolFrame(obj, t_idx: integer, new_frm_tt: vector)
+toolFrame(out wf_H_tt: matrix, obj, wf_R_b: matrix, wf_p_b: vector, q_j: vector, t_idx: integer)
+toolFrame(out wf_H_tt: matrix, obj, wf_R_b: matrix, wf_p_b: vector, q_j: vector)
+toolFrame(out wf_H_tt: matrix, obj, t_idx: integer)
+toolFrame(out wf_H_tt: matrix, obj)
+jacobianTool(out J_tt: matrix, obj, wf_R_b: matrix, wf_p_b: vector, q_j: vector, t_idx: integer)
+jacobianTool(out J_tt: matrix, obj, wf_R_b: matrix, wf_p_b: vector, q_j: vector)
+jacobianTool(out J_tt: matrix, obj, t_idx: integer)
+jacobianTool(out J_tt: matrix, obj)
+getStateChains(out chn_q: cell, out chn_dq: cell, obj, chain_names: string[1..*])
+getStateChains(out chn_q: cell, out chn_dq: cell, obj, chain_names: string[1..*], q_j: vector, dq_j: vector)
+getStateJointNames(out jnt_q: vector, out jnt_dq: vector, obj, joint_names: string[1..*])
+getStateJointNames(out jnt_q: vector, out jnt_dq: vector, obj, joint_names: string[1..*], q_j: vector, dq_j: vector)
+getStateJointIdx(out jnt_q: vector, out jnt_dq: vector, obj, joint_idx: integer[1..*])
+getStateJointIdx(out jnt_q: vector, out jnt_dq: vector, obj, joint_idx: integer[1..*], q_j: vector, dq_j: vector)
+getStateParams(out stParams: wbmStateParams, obj, stChi: vector)
+getStateParams(out stParams: wbmStateParams, obj, stChi: matrix)
+getPositions(out vqT_b: vector, out q_j: vector, obj, stChi: vector)
+getPositions(out vqT_b: matrix, out q_j: matrix, obj, stChi: matrix)
+getPositionsData(out stmPos: matrix, obj, stmChi: matrix)
+getMixedVelocities(out v_b: vector, out dq_j: vector, obj, stChi: vector)
+getMixedVelocities(out v_b: matrix, out dq_j: matrix, obj, stChi: matrix)
+getBaseVelocities(out v_b: vector, obj, stChi: vector)
+getBaseVelocities(out v_b: matrix, obj, stChi: matrix)
+get.stvChiInit(out stvChi: vector, obj)
+get.stvLen(out stvLen: integer, obj)
+get.vqTInit(out vqT_b: vector, obj)
+get.stvqT(out vqT_b: vector, obj)
+get.robot_body(out robot_body: wbmBody, obj)
+get.robot_config(out robot_config: wbmBaseRobotConfig, obj)
+get.robot_params(out robot_params: wbmBaseRobotParams, obj)
+set.init_state(obj, stInit: wbmStateParams)
+get.init_state(out stInit: wbmStateParams, obj)
+dispConfig(obj, prec: integer)
-initConfig(obj, robot_config: wbmBaseRobotConfig)
-getJointValues(out jnt_q: vector, out jnt_dq: vector, obj, q_j: vector, dq_j: vector, joint_idx: integer[1..*], len: integer)
-checkInitStateDimensions(out result: logical, obj, stInit: wbmStateParams)
-getLinkName(out lnk_name: string, obj, lnk_list: vector, idx: integer)

«use»

«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.
1
+init_state_params

«dataType»
wbmFltgBaseState
+wf_R_b: matrix
+wf_p_b: vector
+wf_v_b: vector

«use»
State parameters of the robot floating base. 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

+config

«dataType»
wbmHumanoidConfig
+jpos_head: vector
+jpos_torso: vector
+jpos_left_arm: vector
+jpos_left_hand: vector
+jpos_left_leg: vector
+jpos_left_feet: vector
+jpos_right_arm: vector
+jpos_right_hand: vector
+jpos_right_leg: vector
+jpos_right_feet: vector

General configuration structure of a humanoid robot.

«dataType»
wbmBaseRobotConfig
+nCstrs: integer
+cstr_link_names: cell
+nPlds: integer
+nTools: integer
+stvLen: integer

Initial condition of a floating base robot.
1
#mwbm_config
1
+robot_config

«dataType»
wbmPayloadLink
+urdf_link_name: string
+pt_mass: double
+lnk_p_pl: vector

0..*
+payload_links

Payload at a specified link, e.g. hands, with a point mass at a specifiied position in the frame of the reference link.

«dataType»
wbmToolLink
+urdf_link_name: string
+ee_vqT_tt: vector

0..*
+tool_links

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

+body
Configuration data to define the body components of the robot.
0..1

«use»

«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)

«use»

«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}
+MAX_JOINT_LIMIT: string {readOnly}
+MAX_NUM_LIMIT: string {readOnly}
+NAME_NOT_EXIST: string {readOnly}
+NOT_HOMOG_MAT: string {readOnly}
+STRING_MISMATCH: string {readOnly}
+UNKNOWN_AXIS_SEQ: string {readOnly}
+UNKNOWN_EXC: string {readOnly}
+SINGULAR_MAT: string {readOnly}
+UNKNOWN_JNT_TYPE: string {readOnly}
+UNKNOWN_LNK_NAME: string {readOnly}
+VALUE_IS_INIT: string {readOnly}
+VALUE_LTE_ZERO: string {readOnly}
+WRONG_ARG: string {readOnly}
+WRONG_ARR_SIZE: string {readOnly}
+WRONG_DATA_TYPE: string {readOnly}
+WRONG_MAT_DIM: string {readOnly}
+WRONG_VEC_DIM: string {readOnly}
+WRONG_VEC_SIZE: string {readOnly}

«use»

«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

«use»

Robot.YourRobotName_ModelType

«dataType»
yourOwnSimConfig

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

«dataType»
genericSimConfig
+DF_GROUND_COLOR_2: vector {readOnly}
+genericSimConfig(out obj: genericSimConfig, main_title: string, robot_sim_body: wbmSimBody, env_settings: wbmSimEnvironment)