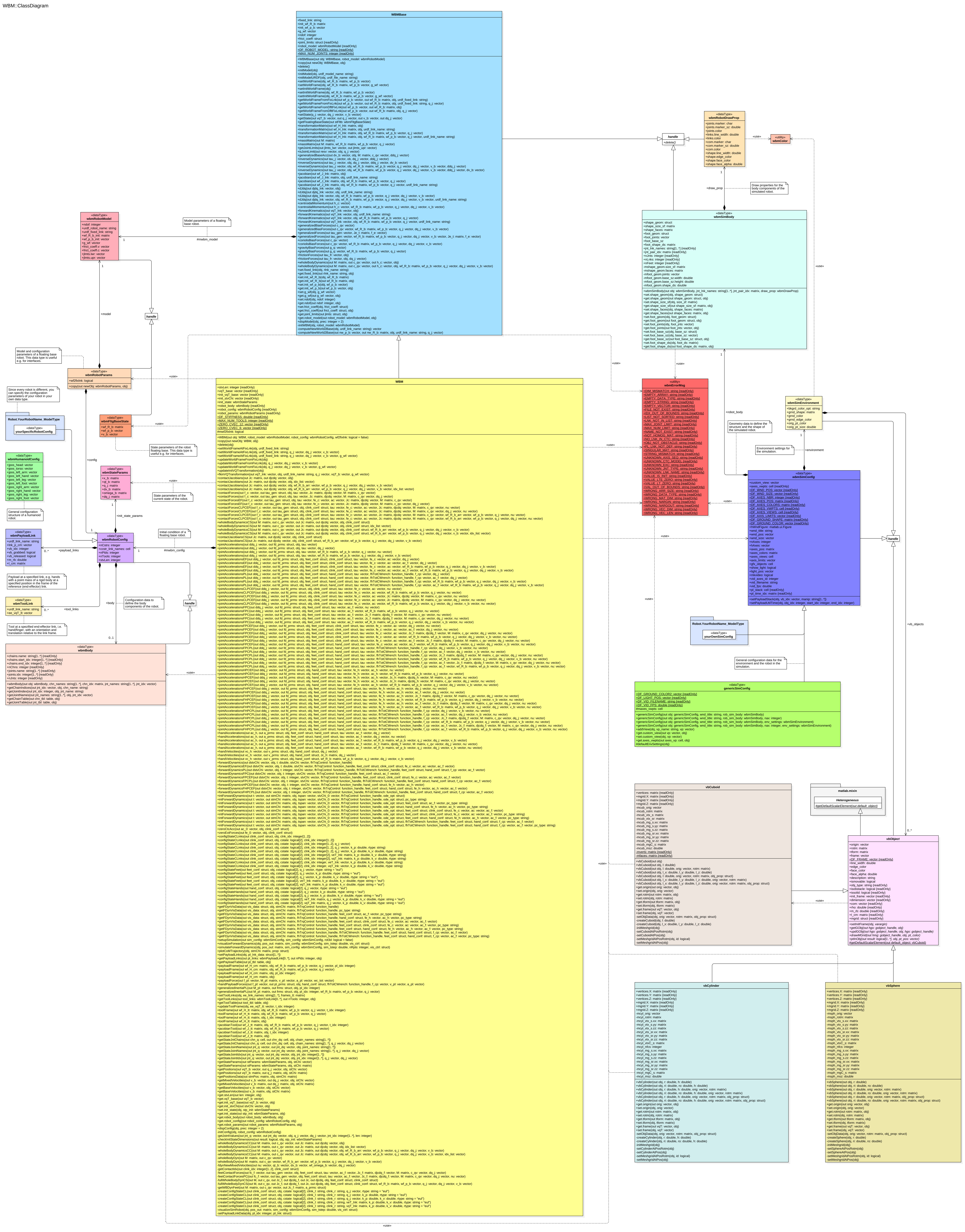
WBM::ClassDiagram
WBMBase
+fixed_link: string
+init_wf_R_b: matrix
+init_wf_p_b: vector
+g_wf: vector
+ndof: integer
+frict_coeff: struct
+joint_limits: struct {readOnly}
+robot_model: wbmRobotModel {readOnly}
+DF_ROBOT_MODEL: string {readOnly}
+MAX_NUM_JOINTS: integer {readOnly}
+WBMBase(out obj: WBMBase, robot_model: wbmRobotModel)
+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)
+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)
+massMatrix(out M: matrix, wf_R_b: matrix, wf_p_b: vector, q_j: vector)
+getJointLimits(out jlmts_lwr: vector, out jlmts_upr: vector)
+isJointLimit(out resv: vector, obj, q_j: vector)
+generalizedBaseAcc(out dv_b: vector, obj, M: matrix, c_qv: vector, ddq_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 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: 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 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, wf_R_b: matrix, wf_p_b: vector, q_j: vector, dq_j: vector, v_b: vector)
+generalizedForces(out tau_gen: vector, Je_t: matrix, f_e: vector)
+generalizedForces(out tau_gen: vector, wf_R_b: matrix, wf_p_b: vector, q_j: vector, dq_j: vector, v_b: vector, Je_t: matrix, f_e: vector)
+coriolisBiasForces(out c_qv: vector)
+coriolisBiasForces(out c_qv: vector, 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, 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)
+get.init_wf_R_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 jlmts: struct, obj)
+get.robot_model(out robot_model: wbmRobotModel, obj)
+dispModel(obj, prec: integer = 2)
-initWBM(obj, robot_model: wbmRobotModel)
-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)
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.
«use»
Model parameters of a floating base robot.
«dataType»
wbmRobotModel
+ndof: integer
+urdf_robot_name: string
+urdf_fixed_link: string
+wf_R_b_init: matrix
+wf_p_b_init: vector
+g_wf: vector
+frict_coeff.v: vector
+frict_coeff.c: vector
+jlmts.lwr: vector
+jlmts.upr: vector
1
#mwbm_model
1
+model
handle
«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
+jnt_lnk_names: string[1..*] {readOnly}
+jnt_pair_idx: matrix {readOnly}
+nJnts: integer {readOnly}
+nLnks: integer {readOnly}
+nFeet: 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, jnt_lnk_names: string[1..*], jnt_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)
Model and configuration parameters of a floating base robot. This data type is useful e.g. for interfaces.
«use»
«use»
«dataType»
wbmRobotParams
+wf2fixlnk: logical
+copy(out newObj: wbmRobotParams, obj)
«use»
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»
wbmFltgBaseState
+wf_R_b: matrix
+wf_p_b: vector
+v_b: vector
«use»
State parameters of the robot floating base. This data type is useful e.g. for interfaces.
«dataType»
wbmHumanoidConfig
+jpos_head: vector
+jpos_torso: vector
+jpos_left_arm: vector
+jpos_left_hand: vector
+jpos_left_leg: vector
+jpos_left_foot: vector
+jpos_right_arm: vector
+jpos_right_hand: vector
+jpos_right_leg: vector
+jpos_right_foot: vector
+config
«dataType»
wbmStateParams
+x_b: matrix
+qt_b: matrix
+q_j: matrix
+dx_b: matrix
+omega_b: matrix
+dq_j: matrix
«use»
State parameters of the current state of the robot.
General configuration structure of a humanoid robot.
1
+init_state_params
1
«dataType»
wbmPayloadLink
+urdf_link_name: string
+lnk_p_cm: vector
+vb_idx: integer
+vb_grabbed: logical
+vb_released: logical
+m_rb: double
+I_cm: matrix
0..*
+payload_links
«dataType»
wbmRobotConfig
+nCstrs: integer
+ccstr_link_names: cell
+nPlds: integer
+nTools: integer
+stvLen: integer
1
Initial condition of a floating base robot.
#mwbm_config
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
0..*
+tool_links
+body
Configuration data to define the body components of the robot.
handle
Tool at a spezified end-effector link, i.e. hand/finger, with an orientation and translation relative to the link frame.
0..1
«dataType»
wbmBody
+chains.name: string[1..*] {readOnly}
+chains.start_idx: integer[1..*] {readOnly}
+chains.end_idx: integer[1..*] {readOnly}
+nChns: integer {readOnly}
+joints.name: string[1..*] {readOnly}
+joints.idx: integer[1..*] {readOnly}
+nJnts: integer {readOnly}
+wbmBody(out obj: wbmBody, chn_names: string[1..*], chn_idx: matrix, jnt_names: string[1..*], jnt_idx: vector)
+getChainIndices(out jnt_idx: vector, obj, chn_name: string)
+getJointIndex(out jnt_idx: integer, obj, jnt_name: string)
+getJointNames(out jnt_names: string[1..*], obj, jnt_idx: vector)
+getChainTable(out chn_tbl: table, obj)
+getJointTable(out jnt_tbl: table, obj)
«use»
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: wbmRobotConfig {readOnly}
+robot_params: wbmRobotParams {readOnly}
+DF_STIFFNESS: double {readOnly}
+MAX_NUM_TOOLS: integer {readOnly}
+ZERO_CVEC_12: vector {readOnly}
+ZERO_CVEC_6: vector {readOnly}
#mwf2fixlnk: logical
+WBM(out obj: WBM, robot_model: wbmRobotModel, robot_config: wbmRobotConfig, wf2fixlnk: logical = false)
+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, dq_j: vector, v_b: 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)
+contactJacobians(out Jc: matrix, out djcdq: vector, obj, idx_list: 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)
+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)
+contactForces(out f_c: vector, out tau_gen: struct, obj, tau: vector, Jc: matrix, djcdq: vector, M: matrix, c_qv: vector, dq_j: vector)
+contactForcesEF(out f_c: vector, out tau_gen: struct, obj, tau: vector, fe_c: vector, ac: vector, Jc: matrix, djcdq: vector, M: matrix, c_qv: vector)
+contactForcesEF(out f_c: vector, out tau_gen: struct, obj, tau: vector, fe_c: vector, ac: vector, Jc: matrix, djcdq: vector, M: matrix, c_qv: vector, dq_j: vector)
+contactForcesCLPCEF(out f_c: vector, out tau_gen: struct, obj, clink_conf: struct, tau: vector, fe_c: vector, ac: vector, Jc: matrix, djcdq: vector, M: matrix, c_qv: vector, nu: vector)
+contactForcesCLPCEF(out f_c: vector, out tau_gen: struct, obj, clink_conf: struct, tau: vector, fe_c: vector, ac: vector, Jc: matrix, djcdq: vector, M: matrix, c_qv: vector, dq_j: vector, nu: vector)
+contactForcesCLPCEF(out f_c: vector, out tau_gen: struct, obj, clink_conf: struct, tau: vector, fe_c: vector, ac: vector, Jc: matrix, djcdq: vector, M: matrix, c_qv: vector, wf_R_b_arr: vector, wf_p_b: vector, q_j: vector, nu: vector)
+contactForcesCLPCEF(out f_c: vector, out tau_gen: struct, obj, clink_conf: struct, tau: vector, fe_c: vector, ac: vector, Jc: matrix, djcdq: vector, M: matrix, c_qv: vector, wf_R_b_arr: vector, wf_p_b: vector, q_j: vector, dq_j: vector, nu: vector)
+wholeBodyDynamicsCS(out M: matrix, out c_qv: vector, out Jc: matrix, out djcdq: vector, obj, clink_conf: struct)
+wholeBodyDynamicsCS(out M: matrix, out c_qv: vector, out Jc: matrix, out djcdq: vector, obj, clink_conf: struct, idx_list: vector)
+wholeBodyDynamicsCS(out M: matrix, out c_qv: vector, out Jc: matrix, out djcdq: vector, obj, clink_conf: struct, wf_R_b_arr: vector, wf_p_b: vector, q_j: vector, dq_j: vector, v_b: vector)
+wholeBodyDynamicsCS(out M: matrix, out c_qv: vector, out Jc: matrix, out djcdq: vector, obj, clink_conf: struct, wf_R_b_arr: vector, wf_p_b: vector, q_j: vector, dq_j: vector, v_b: vector, idx_list: vector)
+contactJacobiansCS(out Jc: matrix, out djcdq: vector, obj, clink_conf: struct)
+contactJacobiansCS(out Jc: matrix, out djcdq: vector, obj, clink_conf: struct, wf_R_b_arr: vector, wf_p_b: vector, q_j: vector, dq_j: vector, v_b: vector)
+jointAccelerations(out ddq_j: vector, out fd_prms: struct, obj, tau: vector)
+jointAccelerations(out ddq_j: vector, out fd_prms: struct, obj, tau: vector, dq_j: vector)
+jointAccelerations(out ddq_j: vector, out fd_prms: struct, obj, tau: vector, wf_R_b: matrix, wf_p_b: vector, q_j: vector, nu: vector)
+jointAccelerations(out ddq_j: vector, out fd_prms: struct, obj, tau: vector, wf_R_b: matrix, wf_p_b: vector, q_j: vector, dq_j: vector, v_b: vector)
+jointAccelerationsEF(out ddq_j: vector, out fd_prms: struct, obj, feet_conf: struct, clink_conf: struct, tau: vector, fe_c: vector, ac: vector, dq_j: vector)
+jointAccelerationsEF(out ddq_j: vector, out fd_prms: struct, obj, feet_conf: struct, clink_conf: struct, tau: vector, fe_c: vector, ac: vector, ac_f: vector, dq_j: vector)
+jointAccelerationsEF(out ddq_j: vector, out fd_prms: struct, obj, feet_conf: struct, clink_conf: struct, tau: vector, fe_c: vector, ac: vector, wf_R_b: matrix, wf_p_b: vector, q_j: vector, dq_j: vector, v_b: vector)
+jointAccelerationsEF(out ddq_j: vector, out fd_prms: struct, obj, feet_conf: struct, clink_conf: struct, tau: vector, fe_c: vector, ac: vector, ac_f: vector, wf_R_b: matrix, wf_p_b: vector, q_j: vector, dq_j: vector, v_b: vector)
+jointAccelerationsPL(out ddq_j: vector, out fd_prms: struct, obj, feet_conf: struct, hand_conf: struct, tau: vector, fhTotCWrench: function_handle, f_cp: vector, dq_j: vector)
+jointAccelerationsPL(out ddq_j: vector, out fd_prms: struct, obj, feet_conf: struct, hand_conf: struct, tau: vector, fhTotCWrench: function_handle, f_cp: vector, ac_f: vector, dq_j: vector)
+jointAccelerationsPL(out ddq_j: vector, out fd_prms: struct, obj, feet_conf: struct, hand_conf: struct, tau: vector, fhTotCWrench: function_handle, f_cp: vector, wf_R_b: matrix, wf_p_b: vector, q_j: vector, dq_j: vector, v_b: vector)
+jointAccelerationsPL(out ddq_j: vector, out fd_prms: struct, obj, feet_conf: struct, hand_conf: struct, tau: vector, fhTotCWrench: function_handle, f_cp: vector, ac_f: vector, wf_R_b: matrix, wf_p_b: vector, q_j: vector, dq_j: vector, v_b: vector)
+jointAccelerationsCLPCEF(out ddq_j: vector, out fd_prms: struct, obj, clink_conf: struct, tau: vector, fe_c: vector, ac: vector, nu: vector)
+jointAccelerationsCLPCEF(out ddq_j: vector, out fd_prms: struct, obj, clink_conf: struct, tau: vector, fe_c: vector, ac: vector, wf_R_b: matrix, wf_p_b: vector, q_j: vector, nu: vector)
+jointAccelerationsCLPCEF(out ddq_j: vector, out fd_prms: struct, obj, clink_conf: struct, tau: vector, fe_c: vector, ac: vector, Jc: matrix, djcdq: vector, M: matrix, c_qv: vector, nu: vector)
+jointAccelerationsCLPCEF(out ddq_j: vector, out fd_prms: struct, obj, clink_conf: struct, tau: vector, fe_c: vector, ac: vector, Jc: matrix, djcdq: vector, M: matrix, c_qv: vector, dq_j: vector, nu: vector)
+jointAccelerationsCLPCEF(out ddq_j: vector, out fd_prms: struct, obj, clink_conf: struct, tau: vector, fe_c: vector, ac: vector, wf_R_b: matrix, wf_p_b: vector, q_j: vector, dq_j: vector, v_b: vector, nu: vector)
+jointAccelerationsFPC(out ddq_j: vector, out fd_prms: struct, obj, feet_conf: struct, tau: vector, ac_f: vector, nu: vector)
+jointAccelerationsFPC(out ddq_j: vector, out fd_prms: struct, obj, feet_conf: struct, tau: vector, ac_f: vector, dq_j: vector, nu: vector)
+jointAccelerationsFPC(out ddq_j: vector, out fd_prms: struct, obj, feet_conf: struct, tau: vector, ac_f: vector, Jc_f: matrix, djcdq_f: vector, M: matrix, c_qv: vector, nu: vector)
+jointAccelerationsFPC(out ddq_j: vector, out fd_prms: struct, obj, feet_conf: struct, tau: vector, ac_f: vector, Jc_f: matrix, djcdq_f: vector, M: matrix, c_qv: vector, dq_j: vector, nu: vector)
+jointAccelerationsFPC(out ddq_j: vector, out fd_prms: struct, obj, feet_conf: struct, tau: vector, ac_f: vector, wf_R_b: matrix, wf_p_b: vector, q_j: vector, dq_j: vector, v_b: vector, nu: vector)
+jointAccelerationsFPCEF(out ddq_j: vector, out fd_prms: struct, obj, feet_conf: struct, clink_conf: struct, tau: vector, fe_c: vector, ac: vector, dq_j: vector, nu: vector)
+jointAccelerationsFPCEF(out ddq_j: vector, out fd_prms: struct, obj, feet_conf: struct, clink_conf: struct, tau: vector, fe_c: vector, ac: vector, ac_f: vector, dq_j: vector, nu: vector)
+jointAccelerationsFPCEF(out ddq_j: vector, out fd_prms: struct, obj, feet_conf: struct, clink_conf: struct, tau: vector, fe_c: vector, ac: vector, Jc_f: matrix, djcdq_f: vector, M: matrix, c_qv: vector, dq_j: vector, nu: vector)
+jointAccelerationsFPCEF(out ddq_j: vector, out fd_prms: struct, obj, feet_conf: struct, clink_conf: struct, tau: vector, fe_c: vector, ac: vector, wf_R_b: matrix, wf_p_b: vector, q_j: vector, dq_j: vector, v_b: vector, nu: vector)
+jointAccelerationsFPCEF(out ddq_j: vector, out fd_prms: struct, obj, feet_conf: struct, clink_conf: struct, tau: vector, fe_c: vector, ac: vector, ac_f: vector, Jc_f: matrix, djcdq_f: vector, M: matrix, c_qv: vector, dq_j: vector, nu: vector)
+jointAccelerationsFPCEF(out ddq_j: vector, out fd_prms: struct, obj, feet_conf: struct, hand_conf: struct, tau: vector, fe_c: vector, ac: vector, ac_f: vector, wf_R_b: matrix, wf_p_b: vector, q_j: vector, dq_j: vector, v_b: vector, nu: vector)
+jointAccelerationsFPCPL(out ddq_j: vector, out fd_prms: struct, obj, feet_conf: struct, hand_conf: struct, tau: vector, fhTotCWrench: function_handle, f_cp: vector, dq_j: vector, v_b: vector, nu: vector)
+jointAccelerationsFPCPL(out ddq_j: vector, out fd_prms: struct, obj, feet_conf: struct, hand_conf: struct, tau: vector, fhTotCWrench: function_handle, f_cp: vector, ac_f: vector, dq_j: vector, v_b: vector, nu: vector)
+jointAccelerationsFPCPL(out ddq_j: vector, out fd_prms: struct, obj, feet_conf: struct, hand_conf: struct, tau: vector, fhTotCWrench: function_handle, f_cp: vector, Jc_f: matrix, djcdq_f: vector, M: matrix, c_qv: vector, dq_j: vector, nu: vector)
+jointAccelerationsFPCPL(out ddq_j: vector, out fd_prms: struct, obj, feet_conf: struct, hand_conf: struct, tau: vector, fhTotCWrench: function_handle, f_cp: vector, wf_R_b: matrix, wf_p_b: vector, q_j: vector, dq_j: vector, v_b: vector, nu: vector)
+jointAccelerationsFPCPL(out ddq_j: vector, out fd_prms: struct, obj, feet_conf: struct, hand_conf: struct, tau: vector, fhTotCWrench: function_handle, f_cp: vector, ac_f: vector, Jc_f: matrix, djcdq_f: vector, M: matrix, c_qv: vector, dq_j: vector, nu: vector)
+jointAccelerationsFPCPL(out ddq_j: vector, out fd_prms: struct, obj, feet_conf: struct, hand_conf: struct, tau: vector, fhTotCWrench: function_handle, f_cp: vector, ac_f: vector, wf_R_b: matrix, wf_p_b: vector, q_j: vector, dq_j: vector, v_b: vector, nu: vector)
+jointAccelerationsHPCEF(out ddq_j: vector, out fd_prms: struct, obj, hand_conf: struct, tau: vector, fe_h: vector, ac_h: vector, nu: vector)
+jointAccelerationsHPCEF(out ddq_j: vector, out fd_prms: struct, obj, hand_conf: struct, tau: vector, fe_h: vector, ac_h: vector, wf_R_b: matrix, wf_p_b: vector, q_j: vector, nu: vector)
+jointAccelerationsHPCEF(out ddq_j: vector, out fd_prms: struct, obj, hand_conf: struct, tau: vector, fe_h: vector, ac_h: vector, Jc_h: matrix, djcdq: vector, M: matrix, c_qv: vector, nu: vector)
+jointAccelerationsHPCEF(out ddq_j: vector, out fd_prms: struct, obj, hand_conf: struct, tau: vector, fe_h: vector, ac_h: vector, Jc_h: matrix, djcdq: vector, M: matrix, c_qv: vector, dq_j: vector, nu: vector)
+jointAccelerationsHPCEF(out ddq_j: vector, out fd_prms: struct, obj, hand_conf: struct, tau: vector, fe_h: vector, ac_h: vector, wf_R_b: matrix, wf_p_b: vector, q_j: vector, dq_j: vector, v_b: vector, nu: vector)
+jointAccelerationsFHPCEF(out ddq_j: vector, out fd_prms: struct, obj, feet_conf: struct, hand_conf: struct, tau: vector, fe_h: vector, ac_h: vector, dq_j: vector, nu: vector)
+jointAccelerationsFHPCEF(out ddq_j: vector, out fd_prms: struct, obj, feet_conf: struct, hand_conf: struct, tau: vector, fe_h: vector, ac_h: vector, ac_f: vector, dq_j: vector, nu: vector)
+jointAccelerationsFHPCEF(out ddq_j: vector, out fd_prms: struct, obj, feet_conf: struct, hand_conf: struct, tau: vector, fe_h: vector, ac_h: vector, Jc_f: matrix, djcdq_f: vector, M: matrix, c_qv: vector, dq_j: vector, nu: vector)
+jointAccelerationsFHPCEF(out ddq_j: vector, out fd_prms: struct, obj, feet_conf: struct, hand_conf: struct, tau: vector, fe_h: vector, ac_h: vector, wf_R_b: matrix, wf_p_b: vector, q_j: vector, dq_j: vector, v_b: vector, nu: vector)
+jointAccelerationsFHPCEF(out ddq_j: vector, out fd_prms: struct, obj, feet_conf: struct, hand_conf: struct, tau: vector, fe_h: vector, ac_h: vector, ac_f: vector, Jc_f: matrix, djcdq_f: vector, M: matrix, c_qv: vector, dq_j: vector, nu: vector)
+jointAccelerationsFHPCEF(out ddq_j: vector, out fd_prms: struct, obj, feet_conf: struct, hand_conf: struct, tau: vector, fe_h: vector, ac_h: vector, ac_f: vector, wf_R_b: matrix, wf_p_b: vector, q_j: vector, dq_j: vector, v_b: vector, nu: vector)
+jointAccelerationsFHPCPL(out ddq_j: vector, out fd_prms: struct, obj, feet_conf: struct, hand_conf: struct, tau: vector, fhTotCWrench: function_handle, f_cp: vector, dq_j: vector, v_b: vector, nu: vector)
+jointAccelerationsFHPCPL(out ddq_j: vector, out fd_prms: struct, obj, feet_conf: struct, hand_conf: struct, tau: vector, fhTotCWrench: function_handle, f_cp: vector, ac_f: vector, dq_j: vector, v_b: vector, nu: vector)
+jointAccelerationsFHPCPL(out ddq_j: vector, out fd_prms: struct, obj, feet_conf: struct, hand_conf: struct, tau: vector, fhTotCWrench: function_handle, f_cp: vector, Jc_f: matrix, djcdq_f: vector, M: matrix, c_qv: vector, dq_j: vector, nu: vector)
+jointAccelerationsFHPCPL(out ddq_j: vector, out fd_prms: struct, obj, feet_conf: struct, hand_conf: struct, tau: vector, fhTotCWrench: function_handle, f_cp: vector, wf_R_b: matrix, wf_p_b: vector, q_j: vector, dq_j: vector, v_b: vector, nu: vector)
+jointAccelerationsFHPCPL(out ddq_j: vector, out fd_prms: struct, obj, feet_conf: struct, hand_conf: struct, tau: vector, fhTotCWrench: function_handle, f_cp: vector, ac_f: vector, Jc_f: matrix, djcdq_f: vector, M: matrix, c_qv: vector, dq_j: vector, nu: vector)
+jointAccelerationsFHPCPL(out ddq_j: vector, out fd_prms: struct, obj, feet_conf: struct, hand_conf: struct, tau: vector, fhTotCWrench: function_handle, f_cp: vector, ac_f: vector, wf_R_b: matrix, wf_p_b: vector, q_j: vector, dq_j: vector, v_b: vector, nu: vector)
+handAccelerations(out ac_h, out a_prms: struct, obj, feet_conf: struct, hand_conf: struct, tau: vector, ac_f: vector, dq_j: vector)
+handAccelerations(out ac_h, out a_prms: struct, obj, feet_conf: struct, hand_conf: struct, tau: vector, ac_f: vector, dq_j: vector, nu: vector)
+handAccelerations(out ac_h, out a_prms: struct, obj, feet_conf: struct, hand_conf: struct, tau: vector, ac_f: vector, wf_R_b: matrix, wf_p_b: vector, q_j: vector, dq_j: vector, v_b: vector)
+handAccelerations(out ac_h, out a_prms: struct, obj, feet_conf: struct, hand_conf: struct, tau: vector, ac_f: vector, Jc_f: matrix, djcdq_f: vector, M: matrix, c_qv: vector, dq_j: vector, nu: vector)
+handAccelerations(out ac_h, out a_prms: struct, obj, feet_conf: struct, hand_conf: struct, tau: vector, ac_f: vector, wf_R_b: matrix, wf_p_b: vector, q_j: vector, dq_j: vector, v_b: vector, nu: vector)
+handVelocities(out vc_h: vector, out v_prms: struct, obj, hand_conf: struct, dq_j: vector)
+handVelocities(out vc_h: vector, out v_prms: struct, obj, hand_conf: struct, Jc_h: matrix, dq_j: vector)
+handVelocities(out vc_h: vector, out v_prms: struct, obj, hand_conf: struct, wf_R_b: matrix, wf_p_b: vector, q_j: vector, dq_j: vector, v_b: vector)
+forwardDynamics(out dstvChi: vector, obj, t: double, stvChi: vector, fhTrqControl: function_handle)
+forwardDynamicsEF(out dstvChi: vector, obj, t: double, stvChi: vector, fhTrqControl: function_handle, feet_conf: struct, clink_conf: struct, fe_c: vector, ac: vector, ac_f: vector)
+forwardDynamicsPL(out dstvChi: vector, obj, t: integer, stvChi: vector, fhTrqControl: function_handle, fhTotCWrench: function_handle, feet_conf: struct, hand_conf: struct, f_cp: vector, ac_f: vector)
+forwardDynamicsFPC(out dstvChi: vector, obj, t: integer, stvChi: vector, fhTrqControl: function_handle, feet_conf: struct, ac_f: vector)
+forwardDynamicsFPCEF(out dstvChi: vector, obj, t: integer, stvChi: vector, fhTrqControl: function_handle, feet_conf: struct, clink_conf: struct, fe_c: vector, ac: vector, ac_f: vector)
+forwardDynamicsFPCPL(out dstvChi: vector, obj, t: integer, stvChi: vector, fhTrqControl: function_handle, fhTotCWrench: function_handle, feet_conf: struct, hand_conf: struct, f_cp: vector, ac_f: vector)
+forwardDynamicsHPCEF(out dstvChi: vector, obj, t: integer, stvChi: vector, fhTrqControl: function_handle, hand_conf: struct, fe_h: vector, ac_h: vector)
+forwardDynamicsFHPCEF(out dstvChi: vector, obj, t: integer, stvChi: vector, fhTrqControl: function_handle, feet_conf: struct, hand_conf: struct, fe_h: vector, ac_h: vector, ac_f: vector)
+forwardDynamicsFHPCPL(out dstvChi: vector, obj, t: integer, stvChi: vector, fhTrqControl: function_handle, fhTotCWrench: function_handle, feet_conf: struct, hand_conf: struct, f_cp: vector, ac_f: vector)
+intForwardDynamics(out t: vector, out stmChi: matrix, obj, tspan: vector, stvChi_0: vector, fhTrqControl: function_handle, ode_opt: struct)
+intForwardDynamics(out t: vector, out stmChi: matrix, obj, tspan: vector, stvChi_0: vector, fhTrqControl: function_handle, ode_opt: struct, pc_type: string)
+intForwardDynamics(out t: vector, out stmChi: matrix, obj, tspan: vector, stvChi_0: vector, fhTrqControl: function_handle, ode_opt: struct, feet_conf: struct, ac_f: vector, pc_type: string)
+intForwardDynamics(out t: vector, out stmChi: matrix, obj, tspan: vector, stvChi_0: vector, fhTrqControl: function_handle, ode_opt: struct, hand_conf: struct, fe_h: vector, ac_h: vector, pc_type: string)
+intForwardDynamics(out t: vector, out stmChi: matrix, obj, tspan: vector, stvChi_0: vector, fhTrqControl: function_handle, ode_opt: struct, feet_conf: struct, clink_conf: struct, fe_c: vector, ac: vector, ac_f: vector)
+intForwardDynamics(out t: vector, out stmChi: matrix, obj, tspan: vector, stvChi_0: vector, fhTrqControl: function_handle, ode_opt: struct, feet_conf: struct, clink_conf: struct, fe_c: vector, ac: vector, ac_f: vector, pc_type: string)
+intForwardDynamics(out t: vector, out stmChi: matrix, obj, tspan: vector, stvChi_0: vector, fhTrqControl: function_handle, ode_opt: struct, feet_conf: struct, hand_conf: struct, fe_h: vector, ac_h: vector, ac_f: vector, pc_type: string)
+intForwardDynamics(out t: vector, out stmChi: matrix, obj, tspan: vector, stvChi_0: vector, fhTrqControl: function_handle, ode_opt: struct, fhTotCWrench: function_handle, feet_conf: struct, hand_conf: struct, f_cp: vector, ac_f: vector)
+intForwardDynamics(out t: vector, out stmChi: matrix, obj, tspan: vector, stvChi_0: vector, fhTrqControl: function_handle, ode_opt: struct, fhTotCWrench: function_handle, feet_conf: struct, hand_conf: struct, f_cp: vector, ac_f: vector, pc_type: string)
+zeroCtcAcc(out ac_0: vector, obj, clink_conf: struct)
+zeroExtForces(out fe_0: vector, obj, clink_conf: struct)
+configStateCLinks(out clink_conf: struct, obj, clnk_idx: integer[1..2])
+configStateCLinks(out clink_conf: struct, obj, cstate: logical[2], clnk_idx: integer[1..2])
+configStateCLinks(out clink_conf: struct, obj, cstate: logical[2], clnk_idx: integer[1..2], q_j: vector)
+configStateCLinks(out clink_conf: struct, obj, cstate: logical[2], clnk_idx: integer[1..2], q_j: vector, k_p: double, rtype: string)
+configStateCLinks(out clink_conf: struct, obj, cstate: logical[2], clnk_idx: integer[1..2], q_j: vector, k_p: double, k_v: double, rtype: string)
+configStateCLinks(out clink_conf: struct, obj, cstate: logical[2], clnk_idx: integer[2], veT_lnk: matrix, k_p: double, k_v: double, rtype: string)
+configStateCLinks(out clink_conf: struct, obj, cstate: logical[2], clnk_idx: integer[2], vqT_lnk: matrix, k_p: double, k_v: double, rtype: string)
+configStateCLinks(out clink_conf: struct, obj, cstate: logical[2], clnk_idx: integer, veT_lnk: vector, k_p: double, k_v: double, rtype: string)
+configStateCLinks(out clink_conf: struct, obj, cstate: logical[2], clnk_idx: integer, vqT_lnk: vector, k_p: double, k_v: double, rtype: string)
+configStateFeet(out feet_conf: struct, obj, cstate: logical[2], q_j: vector, rtype: string = "eul")
+configStateFeet(out feet_conf: struct, obj, cstate: logical[2], q_j: vector, k_p: double, rtype: string = "eul")
+configStateFeet(out feet_conf: struct, obj, cstate: logical[2], q_j: vector, k_p: double, k_v: double, rtype: string = "eul")
+configStateFeet(out feet_conf: struct, obj, cstate: logical[2], veT_lnk: matrix, k_p: double, k_v: double, rtype: string = "eul")
+configStateFeet(out feet_conf: struct, obj, cstate: logical[2], vqT_lnk: matrix, k_p: double, k_v: double, rtype: string = "eul")
+configStateHands(out hand_conf: struct, obj, cstate: logical[2], q_j: vector, rtype: string = "eul")
+configStateHands(out hand_conf: struct, obj, cstate: logical[2], q_j: vector, k_p: double, rtype: string = "eul")
+configStateHands(out hand_conf: struct, obj, cstate: logical[2], q_j: vector, k_p: double, k_v: double, rtype: string = "eul")
+configStateHands(out hand_conf: struct, obj, cstate: logical[2], veT_lnk: matrix, q_j: vector, k_p: double, k_v: double, rtype: string = "eul")
+configStateHands(out hand_conf: struct, obj, cstate: logical[2], vqT_lnk: matrix, q_j: vector, k_p: double, k_v: double, rtype: string = "eul")
+getFDynVisData(out vis_data: struct, obj, stmChi: matrix, fhTrqControl: function_handle)
+getFDynVisData(out vis_data: struct, obj, stmChi: matrix, fhTrqControl: function_handle, pc_type: string)
+getFDynVisData(out vis_data: struct, obj, stmChi: matrix, fhTrqControl: function_handle, feet_conf: struct, ac_f: vector, pc_type: string)
+getFDynVisData(out vis_data: struct, obj, stmChi: matrix, fhTrqControl: function_handle, hand_conf: struct, fe_h: vector, ac_h: vector, pc_type: string)
+getFDynVisData(out vis_data: struct, obj, stmChi: matrix, fhTrqControl: function_handle, feet_conf: struct, clink_conf: struct, fe_c: vector, ac: vector, ac_f: vector)
+getFDynVisData(out vis_data: struct, obj, stmChi: matrix, fhTrqControl: function_handle, feet_conf: struct, clink_conf: struct, fe_c: vector, ac: vector, ac_f: vector, pc_type: string)
+getFDynVisData(out vis_data: struct, obj, stmChi: matrix, fhTrqControl: function_handle, feet_conf: struct, hand_conf: struct, fe_h: vector, ac_h: vector, ac_f: vector, pc_type: string)
+getFDynVisData(out vis_data: struct, obj, stmChi: matrix, fhTrqControl: function_handle, fhTotCWrench: function_handle, feet_conf: struct, hand_conf: struct, f_cp: vector, ac_f: vector)
+getFDynVisData(out vis_data: struct, obj, stmChi: matrix, fhTrqControl: function_handle, fhTotCWrench: function_handle, feet_conf: struct, hand_conf: struct, f_cp: vector, ac_f: vector, pc_type: string)
+setupSimulation(out sim_config: wbmSimConfig, sim_config: wbmSimConfig, rot3d: logical = false)
+visualizeForwardDynamics(obj, pos_out: matrix, sim_config: wbmSimConfig, sim_tstep: double, vis_ctrl: struct)
+simulateForwardDynamics(obj, pos_out: matrix, sim_config: wbmSimConfig, sim_tstep: double, nRpts: integer, vis_ctrl: struct)
+plotCoMTrajectory(obj, stmChi: matrix, prop: struct)
+setPayloadLinks(obj, pl_lnk_data: struct[1..*])
+getPayloadLinks(out pl_links: wbmPayloadLink[0..*], out nPlds: integer, obj)
+getPayloadTable(out pl_tbl: table, obj)
+payloadFrame(out wf_H_cm: matrix, obj, wf_R_b: matrix, wf_p_b: vector, q_j: vector, pl_idx: integer)
+payloadFrame(out wf_H_cm: matrix, obj, wf_R_b: matrix, wf_p_b: vector, q_j: vector)
+payloadFrame(out wf_H_cm: matrix, obj, pl_idx: integer)
+payloadFrame(out wf_H_cm: matrix, obj)
+payloadForce(out f_pl: vector, M_pl: matrix, v_pl: vector, a_pl: vector, wc_tot: vector)
+handPayloadForces(out f_pl: vector, out pl_prms: struct, obj, hand_conf: struct, fhTotCWrench: function_handle, f_cp: vector, v_pl: vector, a_pl: vector)
+generalizedInertiaPL(out M_pl: matrix, out frms: struct, obj, pl_idx: integer)
+generalizedInertiaPL(out M_pl: matrix, out frms: struct, obj, pl_idx: integer, wf_R_b: matrix, wf_p_b: vector, q_j: vector)
+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, ee_vqT_tt: vector, t_idx: integer)
+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 wf_J_tt: matrix, obj, wf_R_b: matrix, wf_p_b: vector, q_j: vector, t_idx: integer)
+jacobianTool(out wf_J_tt: matrix, obj, wf_R_b: matrix, wf_p_b: vector, q_j: vector)
+jacobianTool(out wf_J_tt: matrix, obj, t_idx: integer)
+jacobianTool(out wf_J_tt: matrix, obj)
+getStateJntChains(out chn_q: cell, out chn_dq: cell, obj, chain_names: string[1..*])
+getStateJntChains(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, jnt_idx: integer[1..*])
+getStateJointIdx(out jnt_q: vector, out jnt_dq: vector, obj, jnt_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.stvLen(out len: integer, obj)
+get.vqT_base(out vqT_b: vector)
+get.init_vqT_base(out vqT_b: vector, obj)
+get.init_stvChi(out stvChi: vector, obj)
+set.init_state(obj, stp_init: wbmStateParams)
+get.init_state(out stp_init: wbmStateParams, obj)
+get.robot_body(out robot_body: wbmBody, obj)
+get.robot_config(out robot_config: wbmRobotConfig, obj)
+get.robot_params(out robot_params: wbmRobotParams, obj)
+dispConfig(obj, prec: integer = 2)
-initConfig(obj, robot_config: wbmRobotConfig)
-getJointValues(out jnt_q: vector, out jnt_dq: vector, obj, q_j: vector, dq_j: vector, jnt_idx: integer[1..*], len: integer)
-checkInitStateDimensions(out result: logical, obj, stp_init: wbmStateParams)
-wholeBodyDynamicsCC(out M: matrix, out c_qv: vector, out Jc: matrix, out djcdq: vector, obj)
-wholeBodyDynamicsCC(out M: matrix, out c_qv: vector, out Jc: matrix, out djcdq: vector, obj, idx_list: vector)
-wholeBodyDynamicsCC(out M: matrix, out c_qv: vector, 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)
-wholeBodyDynamicsCC(out M: matrix, out c_qv: vector, 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)
-wholeBodyDyn(out M: matrix, out c_qv: vector)
-wholeBodyDyn(out M: matrix, out c_qv: vector, wf_R_b_arr: vector, wf_p_b: vector, q_j: vector, dq_j: vector, v_b: vector)
-fdynNewMixedVelocities(out nu: vector, qt_b: vector, dx_b: vector, wf_omega_b: vector, dq_j: vector)
-getContactIdx(out clink_idx: integer[1..2], clink_conf: struct)
-feetContactForces(out fc_f: vector, out tau_gen: vector, obj, feet_conf: struct, tau: vector, ac_f: vector, Jc_f: matrix, djcdq_f: vector, M: matrix, c_qv: vector, dq_j: vector)
-feetContactForcesPC(out fc_f: vector, out tau_gen: vector, obj, feet_conf: struct, tau: vector, ac_f: vector, Jc_f: matrix, djcdq_f: vector, M: matrix, c_qv: vector, dq_j: vector, nu: vector)
-fullWholeBodyDynCS(out M, out c_qv, out Jc_f, out djcdq_f, out Jc, out djcdq, obj, feet_conf: struct, clink_conf: struct)
-fullWholeBodyDynCS(out M, out c_qv, out Jc_f, out djcdq_f, out Jc, out djcdq, obj, feet_conf: struct, clink_conf: struct, wf_R_b: matrix, wf_p_b: vector, q_j: vector, dq_j: vector, v_b: vector)
-getWBDynFeet(out M: matrix, out c_qv: vector, out Jc_f: matrix, a_prms: struct)
-createConfigStateCL(out clink_conf: struct, obj, cstate: logical[2], clink_l: string, clink_r: string, q_j: vector, rtype: string = "eul")
-createConfigStateCL(out clink_conf: struct, obj, cstate: logical[2], clink_l: string, clink_r: string, q_j: vector, k_p: double, rtype: string = "eul")
-createConfigStateCL(out clink_conf: struct, obj, cstate: logical[2], clink_l: string, clink_r: string, q_j: vector, k_p: double, k_v: double, rtype: string = "eul")
-createConfigStateCL(out clink_conf: struct, obj, cstate: logical[2], clink_l: string, clink_r: string, veT_lnk: matrix, k_p: double, k_v: double, rtype: string = "eul")
-createConfigStateCL(out clink_conf: struct, obj, cstate: logical[2], clink_l: string, clink_r: string, vqT_lnk: matrix, k_p: double, k_v: double, rtype: string = "eul")
-visualizeSimRobot(obj, pos_out: matrix, sim_config: wbmSimConfig, sim_tstep: double, vis_ctrl: struct)
-setPayloadLinkData(obj, pl_idx: integer, pl_lnk: struct)
«use»
«use»
«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}
+LIST_NOT_SORTED: string {readOnly}
+LNK_NOT_IN_LIST: string {readOnly}
+MAX_JOINT_LIMIT: string {readOnly}
+MAX_NUM_LIMIT: string {readOnly}
+NAME_NOT_EXIST: string {readOnly}
+NOT_HOMOG_MAT: string {readOnly}
+NO_LNK_IN_CTC: string {readOnly}
+OBJ_NOT_OBSTACLE: string {readOnly}
+PL_LNK_NOT_DEF: string {readOnly}
+SINGULAR_MAT: string {readOnly}
+STRING_MISMATCH: string {readOnly}
+UNKNOWN_AXIS_SEQ: string {readOnly}
+UNKNOWN_CTC_MODEL {readOnly}
+UNKNOWN_EXC: string {readOnly}
+UNKNOWN_JNT_TYPE: string {readOnly}
+UNKNOWN_LNK_NAME: string {readOnly}
+VALUE_IS_INIT: string {readOnly}
+VALUE_LTE_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_NARGOUT: string {readOnly}
+WRONG_VEC_DIM: string {readOnly}
+WRONG_VEC_LEN: string {readOnly}
1
+robot_body
Geometry data to define the structure and the shape of the simulated robot.
«dataType»
wbmSimEnvironment
+bkgrd_color_opt: string
+grnd_shape: matrix
+grnd_color
+grnd_edge_color
+orig_pt_color
+orig_pt_size: double
1
Environment settings for the simulation.
+environment
«use»
«dataType»
wbmSimConfig
+custom_view: vector
+axes_vwpts: cell {readOnly}
+DF_WND_POS: vector {readOnly}
+DF_WND_SIZE: vector {readOnly}
+DF_AXES_NBR: integer {readOnly}
+DF_AXES_POS: matrix {readOnly}
+DF_AXES_COLORS: matrix {readOnly}
+DF_AXES_VWPTS: cell {readOnly}
+DF_AXES_VIEWS: cell {readOnly}
+DF_AXIS_LIMITS: vector {readOnly}
+DF_GROUND_SHAPE: matrix {readOnly}
+DF_GROUND_COLOR: vector {readOnly}
+hWndFigure: matlab.ui.Figure
+wnd_title: string
+wnd_pos: vector
+wnd_size: vector
+nAxes: integer
+hAxes: vector
+axes_pos: matrix
+axes_colors: matrix
+axes_views: cell
+axis_limits: vector
+gfx_objects: cell
+show_light: logical
+light_pos: vector
+mkvideo: logical
+vid_axes_id: integer
+vid_filename: string
+vid_fps: double
+pl_stack: cell {readOnly}
+pl_time_idx: matrix {readOnly}
+setPayloadStack(obj, vb_idx: vector, manip: string[1..*])
+setPayloadUtilTime(obj, obj_idx: integer, start_idx: integer, end_idx: integer)
handle
«use»
+vb_objects
Robot.YourRobotName_ModelType
«dataType»
yourOwnSimConfig
General configuration data for the environment and the robot in the simulaton.
«dataType»
genericSimConfig
+DF_GROUND_COLOR2: vector {readOnly}
+DF_LIGHT_POS: vector {readOnly}
+DF_VID_FILENAME: string {readOnly}
+DF_VID_FPS: double {readOnly}
#maxes_vwpts: cell
+genericSimConfig(out obj: genericSimConfig, wnd_title: string, rob_sim_body: wbmSimBody)
+genericSimConfig(out obj: genericSimConfig, wnd_title: string, rob_sim_body: wbmSimBody, nax: integer)
+genericSimConfig(out obj: genericSimConfig, wnd_title: string, rob_sim_body: wbmSimBody, env_settings: wbmSimEnvironment)
+genericSimConfig(out obj: genericSimConfig, wnd_title: string, rob_sim_body: wbmSimBody, nax: integer, env_settings: wbmSimEnvironment)
+addView(obj, vp_name: string, vp: vector)
+get.custom_view(out vp: vector, obj)
+set.custom_view(obj, vp: vector)
+get.axes_vwpts(out axes_vp: cell, obj)
#defaultEnvSettings(obj)
vbCuboid
+vertices: matrix {readOnly}
+mgrid.X: matrix {readOnly}
+mgrid.Y: matrix {readOnly}
+mgrid.Z: matrix {readOnly}
-mcub_orig: vector
-mcub_rotm: matrix
-mcub_vtx_s: matrix
-mcub_vtx_sr: matrix
-mcub_mg_s.xx: matrix
-mcub_mg_s.yy: matrix
-mcub_mg_s.zz: matrix
-mcub_mg_sr.xx: matrix
-mcub_mg_sr.yy: matrix
-mcub_mg_sr.zz: matrix
-mcub_mgC_s: matrix
-mcub_msz: double
-mverts: matrix {readOnly}
-mfaces: matrix {readOnly}
+vbCuboid(out obj)
+vbCuboid(out obj, l: double)
+vbCuboid(out obj, l: double, orig: vector, rotm: matrix)
+vbCuboid(out obj, l_x: double, l_y: double, l_z: double)
+vbCuboid(out obj, l: double, orig: vector, rotm: matrix, obj_prop: struct)
+vbCuboid(out obj, l_x: double, l_y: double, l_z: double, orig: vector, rotm: matrix)
+vbCuboid(out obj, l_x: double, l_y: double, l_z: double, orig: vector, rotm: matrix, obj_prop: struct)
+get.origin(out orig: vector, obj)
+set.origin(obj, orig: vector)
+get.rotm(out rotm: matrix, obj)
+set.rotm(obj, rotm: matrix)
+get.tform(out tform: matrix, obj)
+set.tform(obj, tform: matrix)
+get.frame(out vqT: vector, obj)
+set.frame(obj, vqT: vector)
-setObjData(obj, orig: vector, rotm: matrix, obj_prop: struct)
-createCuboid(obj, l: double)
-createCuboid(obj, l_x: double, l_y: double, l_z: double)
-initMeshgrid(obj)
-setCuboidAtPosRotm(obj)
-setCuboidAtPos(obj)
-setMeshgridAtPosRotm(obj, id: logical)
-setMeshgridAtPos(obj)
matlab.mixin
Heterogeneous
#getDefaultScalarElement(out default_object)
0..*
vbObject
+origin: vector
+rotm: matrix
+tform: matrix
+frame: vector
+DF_FRAME: vector {readOnly}
+line_width: double
+edge_color
+face_color
+face_alpha: double
+description: string
+ismovable: logical
+obj_type: string {readOnly}
+isobstacle: logical {readOnly}
+issolid: logical {readOnly}
+init_frame: vector {readOnly}
+dimension: vector {readOnly}
+com: vector {readOnly}
+rho: double {readOnly}
+m_rb: double {readOnly}
+I_cm: matrix {readOnly}
+mgrid: struct {readOnly}
+setInitFrame(obj, varargin)
+getGObj(out hgo: gobject_handle, obj)
+updGObj(out hgo: gobject_handle, obj, hgo: gobject_handle)
+drawMGrid(out hmg: gobject_handle, obj, pt_color)
+ptInObj(out result: logical[1..*], obj, pt_pos: vector)
#getDefaultScalarElement(out default_object: vbCuboid)
«use»
«use»
vbCylinder
+vertices.X: matrix {readOnly}
+vertices.Y: matrix {readOnly}
+vertices.Z: matrix {readOnly}
+mgrid.X: matrix {readOnly}
+mgrid.Y: matrix {readOnly}
+mgrid.Z: matrix {readOnly}
-mcyl_orig: vector
-mcyl_rotm: matrix
-mcyl_vtx_s.xx: matrix
-mcyl_vtx_s.yy: matrix
-mcyl_vtx_s.zz: matrix
-mcyl_vtx_sr.xx: matrix
-mcyl_vtx_sr.yy: matrix
-mcyl_vtx_sr.zz: matrix
-mcyl_vtxC_s: matrix
-mcyl_nfcs: integer
-mcyl_mg_s.xx: matrix
-mcyl_mg_s.yy: matrix
-mcyl_mg_s.zz: matrix
-mcyl_mg_sr.xx: matrix
-mcyl_mg_sr.yy: matrix
-mcyl_mg_sr.zz: matrix
-mcyl_mgC_s: matrix
-mcyl_msz: double
+vbCylinder(out obj, r: double, h: double)
+vbCylinder(out obj, ri: double, ro: double, h: double)
+vbCylinder(out obj, r: double, h: double, orig: vector, rotm: matrix)
+vbCylinder(out obj, ri: double, ro: double, h: double, orig: vector, rotm: matrix)
+vbCylinder(out obj, r: double, h: double, orig: vector, rotm: matrix, obj_prop: struct)
+vbCylinder(out obj, ri: double, ro: double, h: double, orig: vector, rotm: matrix, obj_prop: struct)
+get.origin(out orig: vector, obj)
+set.origin(obj, orig: vector)
+get.rotm(out rotm: matrix, obj)
+set.rotm(obj, rotm: matrix)
+get.tform(out tform: matrix, obj)
+set.tform(obj, tform: matrix)
+get.frame(out vqT: vector, obj)
+set.frame(obj, vqT: vector)
-setObjData(obj, orig: vector, rotm: matrix, obj_prop: struct)
-createCylinder(obj, r: double, h: double)
-createCylinder(obj, ri: double, ro: double, h: double)
-initMeshgrid(obj)
-setCylinderAtPosRotm(obj)
-setCylinderAtPos(obj)
-setMeshgridAtPosRotm(obj, id: logical)
-setMeshgridAtPos(obj)
vbSphere
+vertices.X: matrix {readOnly}
+vertices.Y: matrix {readOnly}
+vertices.Z: matrix {readOnly}
+mgrid.X: matrix {readOnly}
+mgrid.Y: matrix {readOnly}
+mgrid.Z: matrix {readOnly}
-msph_orig: vector
-msph_rotm: matrix
-msph_vtx_s.xx: matrix
-msph_vtx_s.yy: matrix
-msph_vtx_s.zz: matrix
-msph_vtx_sr.xx: matrix
-msph_vtx_sr.yy: matrix
-msph_vtx_sr.zz: matrix
-msph_vtxC_s: matrix
-msph_nfcs: integer
-msph_mg_s.xx: matrix
-msph_mg_s.yy: matrix
-msph_mg_s.zz: matrix
-msph_mg_sr.xx: matrix
-msph_mg_sr.yy: matrix
-msph_mg_sr.zz: matrix
-msph_mgC_s: matrix
-msph_msz: double
+vbSphere(out obj, r: double)
+vbSphere(out obj, ri: double, ro: double)
+vbSphere(out obj, r: double, orig: vector, rotm: matrix)
+vbSphere(out obj, ri: double, ro: double, orig: vector, rotm: matrix)
+vbSphere(out obj, r: double, orig: vector, rotm: matrix, obj_prop: struct)
+vbSphere(out obj, ri: double, ro: double, orig: vector, rotm: matrix, obj_prop: struct)
+get.origin(out orig: vector, obj)
+set.origin(obj, orig: vector)
+get.rotm(out rotm: matrix, obj)
+set.rotm(obj, rotm: matrix)
+get.tform(out tform: matrix, obj)
+set.tform(obj, tform: matrix)
+get.frame(out vqT: vector, obj)
+set.frame(obj, vqT: vector)
-setObjData(obj, orig: vector, rotm: matrix, obj_prop: struct)
-createSphere(obj, r: double)
-createSphere(obj, ri: double, ro: double)
-initMeshgrid(obj)
-setSphereAtPosRotm(obj)
-setSphereAtPos(obj)
-setMeshgridAtPosRotm(obj, id: logical)
-setMeshgridAtPos(obj)
«use»