WBM::WBM-ClassDiagram

WBMBase
+urdfLinkName: string
+wf_R_rootLnk: matrix
+wf_p_rootLnk: vector
+g_wf: vector
+robot_model: wbmBaseModelParams

+WBMBase(out obj: WBMBase, model_params: wbmBaseModelParams)
+copy(out newObj: WBMBase, obj)
+delete()
+initModel(obj)
+initModel(obj, urdf_robot_name: string)
+initModelURDF(obj, urdf_file_name: string)
+setWorldFrame(obj, wf_R_rootLnk: matrix, wf_p_rootLnk: vector)
+setWorldFrame(obj, wf_R_rootLnk: matrix, wf_p_rootLnk: vector, g_wf: vector)
+updateWorldFrame(obj)
+updateWorldFrame(obj, wf_R_rootLnk: matrix, wf_p_rootLnk: vector, g_wf: vector)
+getWorldFrameFromFixedLink(out w_p_b: vector, out w_R_b: matrix, obj)
+getWorldFrameFromFixedLink(out w_p_b: vector, out w_R_b: matrix, obj, q_j: vector)
+getWorldFrameFromFixedLink(out w_p_b: vector, out w_R_b: matrix, obj, urdf_link_name: string)
+getWorldFrameFromFixedLink(out w_p_b: vector, out w_R_b: matrix, obj, urdf_link_name: string, 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)
+massMatrix(out M: matrix)
+massMatrix(out M: matrix, wf_R_rootLnk: matrix, wf_p_rootLnk: vector, q_j: vector)
+getJointLimits(out jl_lower: vector, out jl_upper: vector)
+jacobian(out J: matrix, obj)
+jacobian(out J: matrix, obj, urdf_link_name: string)
+jacobian(out J: matrix, obj, wf_R_rootLnk: matrix, wf_p_rootLnk: vector, q_j: vector)
+jacobian(out J: matrix, obj, urdf_link_name: string, wf_R_rootLnk: matrix, wf_p_rootLnk: vector, q_j: vector)
+dJdq(out djdq: vector, obj)
+dJdq(out djdq: vector, obj, urdf_link_name: string)
+dJdq(out djdq: vector, obj, wf_R_rootLnk: matrix, wf_p_rootLnk: vector, q_j: vector, dq_j: vector, v_b: vector)
+dJdq(out djdq: vector, obj, urdf_link_name: string, wf_R_rootLnk: matrix, wf_p_rootLnk: vector, q_j: vector, dq_j: vector, v_b: vector)
+centroidalMomentum(out h_c: vector)
+centroidalMomentum(out h_c: vector, wf_R_rootLnk: matrix, wf_p_rootLnk: vector, q_j: vector, dq_j: vector, v_b: vector)
+forwardKinematics(out wf_vqT_rlnk: vector, obj)
+forwardKinematics(out wf_vqT_rlnk: vector, obj, urdf_link_name: string)
+forwardKinematics(out wf_vqT_rlnk: vector, obj, wf_R_rootLnk: matrix, wf_p_rootLnk: vector, q_j: vector)
+forwardKinematics(out wf_vqT_rlnk: vector, obj, urdf_link_name: string, wf_R_rootLnk: matrix, wf_p_rootLnk: vector, q_j: vector)
+generalizedBiasForces(out C_qv: vector)
+generalizedBiasForces(out C_qv: vector, wf_R_rootLnk: matrix, wf_p_rootLnk: vector, q_j: vector, dq_j: vector, v_b: vector)
+coriolisCentrifugalForces(out tau_c: vector)
+coriolisCentrifugalForces(out tau_c: vector, wf_R_rootLnk: matrix, wf_p_rootLnk: vector, q_j: vector, dq_j: vector, v_b: vector)
+gravityForces(out tau_g: vector)
+gravityForces(out tau_g: vector, wf_R_rootLnk: matrix, wf_p_rootLnk: vector, q_j: vector)
+set.urdfLinkName(obj, new_link_name: string)
+get.urdfLinkName(out link_name: string, obj)
+set.wf_R_rootLnk(obj, wf_R_rlnk: matrix)
+set.wf_p_rootLnk(obj, wf_p_rlnk: vector)
+set.g_wf(obj, g: vector)
+get.robot_model(out robot_model: wbmBaseModelParams, obj)
+dispWBMParams(obj, prec: integer)
-initWBM(obj, model_params: wbmBaseModelParams)
-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» wbmDrawProp
+joints: struct
+links: struct
+com: struct
+shape: struct
+ground_color

«utility» wbmColor

«dataType» wbmBaseModelParams
+urdfRobot: string
+urdfLinkName: string
+wf_R_rootLnk: matrix
+wf_p_rootLnk: vector
+g_wf: vector

WBM base parameters for any model of a floating base 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_names: string[1..*]
+joint_pair_idx: matrix
+nJoints: integer
+nLinks: integer
+nFeets: integer
#mShape_geom: struct
#mFoot_geom: struct

+wbmSimBody(out obj: wbmSimBody, robot_joint_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)

«dataType» wbmBody
+chain_names: string[0..*]
+chain_start_idx: integer[0..*]
+chain_end_idx: integer[0..*]
+nChains: integer
+joint_names: string[0..*]
+joint_idx: integer[0..*]
+nJoints: integer[0..*]

+wbmBody(out obj: wbmBody, chain_names: string[1..*], chain_pos: matrix)
+wbmBody(out obj: wbmBody, chain_names: string[1..*], chain_pos: matrix, joint_names: string[1..*], joint_idx: vector)
+getChainTable(out chn_tbl: table, obj)
+getJointTable(out jnt_tbl: table, obj)

«dataType» wbmStateParams
+x_b: matrix
+qt_b: matrix
+q_j: matrix
+dx_b: matrix
+omega_b: matrix
+dq_j: matrix

State parameters of the robot.

WBM
+stvChiInit: vector
+stvqT: vector
+vqTinit: vector
+stvLen: integer
+robot_config: wbmBaseRobotConfig
+robot_body: wbmBody

+WBM(out obj: WBM, model_params: wbmBaseModelParams, robot_config: wbmBaseRobotConfig)
+WBM(out obj: WBM, model_params: wbmBaseModelParams, robot_config: wbmBaseRobotConfig, wf2FixLnk: logical)
+copy(out newObj: WBM, obj)
+delete(obj)
+setWorldFrame2FixedLink(obj, urdf_link_name: string)
+setWorldFrame2FixedLink(obj, q_j: vector, dq_j: vector, v_b: vector, g_wf: vector, urdf_link_name: string)
+setWorldFrame2FixedLink(obj, q_j: vector, dq_j: vector, v_b: vector, g_wf: vector)
+updateInitRotoTranslation(obj)
+computeFKinRotoTranslation(out wf_vqT_lnkfr: vector, obj, urdf_link_name: string, q_j: vector, vqT: vector, g_wf: vector)
+forwardDynamics(out dstvChi: vector, out h: vector, obj, t: integer, stvChi: vector, ctrlTrqs: struct)
+setupSimulation(sim_config: absSimConfig, sim_config: absSimConfig)
+visualizeForwardDynamics(obj, x_out: matrix, sim_config: absSimConfig, sim_tstep: double, vis_ctrl: struct)
+simulateForwardDynamics(obj, x_out: matrix, sim_config: absSimConfig, sim_tstep: double, nRpts: integer, vis_ctrl: struct)
+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, stvChi: vector)
+getStateParamsData(out stParams: wbmStateParams, obj, chi: matrix)
+getPositions(out stvPos: vector, obj, stvChi: vector)
+getPositionsData(out stmPos: matrix, obj, chi: matrix)
+getVelocities(out stvVel: vector, obj, stvChi: vector)
+getVelocitiesData(out stmVel: matrix, obj, chi: matrix)
+getBaseVelocities(out stvVelb: vector, obj, stvChi: vector)
+getBaseVelocitiesData(out stmVelb: matrix, obj, chi: matrix)
+getRotoTranslation(out vqT: vector, stParams: wbmStateParams)
+toStateVector(out stvChi: vector, stParams: wbmStateParams)
+get.stvChiInit(out stvChiInit: vector, obj)
+get.stvLen(out stvLen: integer, obj)
+get.vqTinit(out vqTinit: vector, obj)
+get.stvqT(out stvqT: vector, obj)
+get.robot_config(out robot_config: wbmBaseRobotConfig, obj)
+get.robot_body(out robot_body: wbmBody, obj)
+dispWBMConfig(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)

Initial condition of any floating base robot.

General configuration structure of a humanoid robot.

«dataType» wbmBaseRobotConfig
+ndof: integer
+nCstrs: integer
+cstrLinkNames: cell
+dampCoeff: double
+stvLen: integer

Robot.YourRobotName
«dataType» yourSpecificRobotConfig

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

«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

«utility» wbmErrorMsg
+DIM_MISMATCH: string {readOnly}
+EMPTY_CELL_ARR: string {readOnly}
+EMPTY_STRING: string {readOnly}
+EMPTY_VECTOR: string {readOnly}
+FILE_NOT_EXIST: string {readOnly}
+STRING_MISMATCH: string {readOnly}
+UNKNOWN_EXC: string {readOnly}
+VALUE_IS_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}

«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
+main_title: string
+main_pos: vector
+hAxes
+axes_pos: matrix
+axes_colors: matrix
+axis_limits: vector
+plot_objs: cell
+ground_shape: matrix
+ground_color: vector
+robot_joint_names: cell {readOnly}
+nJoints: integer {readOnly}
+nLinks: integer {readOnly}

Robot.YourRobotName
«dataType» yourOwnSimConfig

«dataType» genericSimConfig
+genericSimConfig(out obj: genericSimConfig, robot_sim_body: wbmSimBody, main_title: string)