Glossary of XML Tags used in CASPR-

1 Introduction

In CASPR, XML scripting is used for a number of applications including describing models and providing information for the GUI settings. In this document a glossary of terms for each of these applications will be provided.

2 Model Description using XML

A new CDPR model can be added to CASPR by creating the following XML files:

1. **bodies.xml**: The rigid body structure of the CDPM, such as the inertia properties and the type of joints.
2. **cables.xml**: The set of cable arrangements for the cable attachment locations and cable properties.
3. **trajectories.xml**: The set of possible trajectories that the robot may execute.
4. **operational_space.xml** [*optional*]: Defines the operational space for the CDPR.

The sections below provide details on the meaning of xml tags and structures for each of these documents.

2.1 Bodies XML

Code Sample 1 shows an example bodies XML file. The following can be noted about the structure of this file

1. The document is enclosed by the tag `links`.
2. Within the `links` tag there is a tag associated with a single link.
3. The single link is itself further defined by a two tags `physical` which contains information associated with the physical parameters of the link and `parent` which contains information about the interconnections for the link.

Listing 1: Example Bodies XML File.

```
  <?xml version="1.0" encoding="utf-8"?>
<links display_range="-0.25 1.25 -0.25 1.25 -0.25 1.25" view_angle="-37 32">
  <link_rigid num="1" name="Link 1">
    <joint type="PLANAR_XY" q_initial="0 0 0"/>
    <physical>
      <mass>1</mass>
      <com_location>0.0 0.0 0.0</com_location>
      <end_location>0.0 0.0 0.0</end_location>
      <inertia ref="com">
        <Ixx>0.005208333</Ixx>
        <Iyy>0.005208333</Iyy>
        <Izz>0.010416667</Izz>
        <Ixy>0.0</Ixy>
        <Ixz>0.0</Ixz>
        <Iyz>0.0</Iyz>
      </inertia>
    </physical>
    <parent>
      <num>0</num>
      <location>0.0 0.0 0.0</location>
    </parent>
  </link_rigid>
</links>
```

A more detailed glossary of these tags in addition to all other tags used by the file is provided below.

- `links` - This tag acts as a container for the set of all links that describe the mechanism.

  The tag also contains the attributes `display_range` and `view_angle` which describes the plotting range and viewing angle for the mechanism. `display_range` should be provided in the format $[\underline{x}, \overline{x}, \underline{y}, \overline{y}, \underline{z}, \overline{z}]$ where x, y and z represent the associated axes of the base frame. `view_angle` should be provided in the format `[az,el]` where az is the azimuthal angle and el is the elevation.

- link - The physical parameters and interconnection information of a single link are represented using a link level tag.

  Currently the only supported link level tag is `link_rigid`. This tag also has two attributes `num` and `name` which provide numerical and text identifiers for the link.

- `joint` - The type of joint that the rigid link is connected to. This is given as an enum which matches those given in `src/Model/Bodies/Joints/JointType.m`. The joint tag possesses two attributes: `type` and `q_initial` which provide information about the joint type and initial joint pose, respectively.

- `physical` - This tag acts as a container for the physical parameter information associated with a link. This includes `mass`, `com_location`, `end_location` and `inertia`.

- `mass` - The mass of the link (in kg).

- `com_location` - The displacement vector ${}^{i}\mathbf{r}_{P_iG_i}$ (in m). This corresponds to the position of the centre of mass relative to the joint location expressed in the reference frame of the link.

- `end_location` The displacement vector ${}^{i}\mathbf{r}_{P_iE_i}$ (in m). This corresponds to the position of the link endpoint relative to the joint location expressed in the reference frame of the link.

- `inertia` The inertia of the mechanism. The components of the inertia matrix are expressed as follows

$$I_{\texttt{ref}} = \begin{bmatrix} Ixx & Ixy & Ixz \\ Ixy & Iyy & Iyz \\ Ixz & Iyz & Izz \end{bmatrix} \tag{1}$$

  where `ref` is given as "com" or "joint" as required. Each of these individual terms are themselves provided within their own tag.

- `parent` - The parent tag acts as a container for information that describes the parent link of the current link (`num` and `location`)

- `num` - The num tag indicates the parent link by providing a number associated with that link.

- `location` - The location tag contains the vector ${}^{i-1}\mathbf{r}_{P_{i-1}P_i}$. That is the position of the joint relative to the frame of reference of the previous link.

2.2 Cables XML

Code Sample 2 shows an example cables XML file. The following can be noted about the structure of this file

1. The document is enclosed by the tag `cables`.
2. Within the `cables` tag there is a tag associated with each set of cables (`cable_set`.
3. Within the `cable_set` tag there is a tag associated with each cable.
4. The single cable is itself further defined by a two tags `physical` which contains information associated with the physical properties of the cable and `attachments` which contains information about the attachments of the cable.

Listing 2: Example Cables XML File.

```
  <?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE cables [
  <!ATTLIST cable_set id ID #REQUIRED>
]>

<cables default_cable_set="basic">
  <cable_set id="basic">
    <cable_ideal name="cable 1" attachment_reference="com">
      <properties>
```

```
      <force_min>0.1</force_min>
      <force_max>1000</force_max>
    </properties>
    <attachments>
      <attachment>
        <link>0</link>
        <location>0.0 0.0 0.0</location>
      </attachment>
      <attachment>
        <link>1</link>
        <location>-0.125 0 0</location>
      </attachment>
    </attachments>
  </cable_ideal>
  <cable_ideal name="cable 2" attachment_reference="com">
    <properties>
      <force_min>0.1</force_min>
      <force_max>1000</force_max>
    </properties>
    <attachments>
      <attachment>
        <link>0</link>
        <location>1.0 0.0 0.0</location>
      </attachment>
      <attachment>
        <link>1</link>
        <location>0.125 0 0</location>
      </attachment>
    </attachments>
  </cable_ideal>
  <cable_ideal name="cable 3" attachment_reference="com">
    <properties>
      <force_min>0.1</force_min>
      <force_max>1000</force_max>
    </properties>
    <attachments>
      <attachment>
        <link>0</link>
        <location>1.0 1.0 0.0</location>
      </attachment>
      <attachment>
        <link>1</link>
        <location>0.125 0 0</location>
      </attachment>
    </attachments>
  </cable_ideal>
  <cable_ideal name="cable 4" attachment_reference="com">
    <properties>
      <force_min>0.1</force_min>
      <force_max>1000</force_max>
    </properties>
    <attachments>
      <attachment>
        <link>0</link>
        <location>0.0 1.0 0.0</location>
      </attachment>
      <attachment>
        <link>1</link>
        <location>-0.125 0 0</location>
      </attachment>
    </attachments>
  </cable_ideal>
 </cable_set>
</cables>
```

A more detailed glossary of these tags in addition to all other tags used by the file is provided below.

- `cables` - This tag acts as a container for the set of all cables that describe the mechanism.

  The tag also contains the attribute `default_cableset`. This describes which of the cable sets within the xml file should be considered as the default cable set for testing purposes.

- `cable_set` - A cable set represents a set of cable attachments to be simulated. The cable set tag acts as a container for all the cable set information. The tag also contained the attribute `id` which is an identifier for the cable set.

- cable - A cable level tag is a container for different cable information. CASPR currently supports the cable tags `cable_ideal`, `cable_linear_spring`, `cable_vsd_torsion_spring` and `cable_vsd_flexure_linear`. For each cable type this will describe the physical properties of the cable (parameters vary between cable type) and the attachment information.
- `properties` - The properties tag contains all of the physical property information for a cable. This information differs for different cables types as follows.
  - `cable_ideal` - For the ideal cable the properties tag need only contain `force_min` and `force_max` which represent the minimum and maximum force (in N) of the cable.
  - `cable_linear_spring` For this cable type the properties tag contains the tags `force_min`, `force_max` and `K_cable` where `K_cable` represents the cable stiffness (in N/m).
  - `cable_vsd_torsion_spring` - For this cable type the properties tag contains the tags `force_min`, `force_max`, `K_cable`, `num_torsion_springs`, `torsion_spring_stiffness` and `torsion_spring_length`. Here
    `num_torsion_springs` represents the number of torsional springs in the `vsd` unit, `torsion_spring_stiffness` represents the stiffness of the springs (in Nm/rad) and `torsion_spring_length` represents the length of the torsion spring (in m).
  - `cable_vsd_flexure_linear` - For this cable type the properties tag contains the tags `force_min`, `force_max`, `K_cable`,
    `vsd_force_deformation_relation`. Here
    `vsd_force_deformation_relation` represents the relative deformation vector.
- `attachments` - This tag is a container for the attachment information for a cable. It should contain at least two `attachment` tags for the cable (more if the cable is comprised of multiple segments).
- `attachment` - A tag that contains an individual attachment location (`link` and `location`).
- `link` - The link that the attachment connects to (0 for the base).
- `location` The vector ${}^{i}\mathbf{r}_{P_iA_{ijk}}$ which is the location of the attachment in the frame of reference of the link.

2.3 Trajectories XML

Code Sample 3 shows an example trajectories XML file. The following can be noted about the structure of this file

1. The document is enclosed by the tag `trajectories`.
2. Within the `trajectories` tag there is a tag associated with each trajectory (`trajectory`) which contains an attribute of the `time_step`.
3. The trajectory tag is then described by a `points` tag containing a set of `point` tags
4. Each of these tags acts as a container for a set of joint space information given by `q`, `q_dot` and `q_ddot` and also provides information about the associated `time` for the point.

Listing 3: Example Trajectory XML File.

```
  <?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE trajectories [
  <!ATTLIST trajectory id ID #REQUIRED>
]>

<trajectories>
  <trajectory id="x_simple" time_step="0.01">
    <points>
      <point>
        <q>0.3 0.5 0.0</q>
        <q_dot>0.0 0.0 0.0</q_dot>
        <q_ddot>0.0 0.0 0.0</q_ddot>
      </point>
      <point time="1.0">
        <q>0.7 0.5 0.0</q>
        <q_dot>0.0 0.0 0.0</q_dot>
        <q_ddot>0.0 0.0 0.0</q_ddot>
      </point>
    </point>
```

```
  </trajectory>
  <trajectory id="general_1" time_step="0.01">
    <points>
      <point>
        <q>0.3 0.6 0.1</q>
        <q_dot>0.0 0.0 0.0</q_dot>
        <q_ddot>0.0 0.0 0.0</q_ddot>
      </point>
      <point>
        <q>0.7 0.2 0.0</q>
        <q_dot>0.0 0.0 0.0</q_dot>
        <q_ddot>0.0 0.0 0.0</q_ddot>
      </point>
    </points>
  </trajectory>
</trajectories>
```

A more detailed glossary of these tags in addition to all other tags used by the file is provided below.

- `trajectories` - This tag acts as a container for the set of all trajectories that are to be used by the mechanism.
- `trajectory` - A trajectory represents a single trajectory. This includes a set of `point` tags and the attribute `time_step`.
- `points` - A container for `point` tags.
- `point` - The pose information and an associated `time` attribute.
- `q` - A pose information.
- `q_dot` - The derivative of the pose. One parameter should be given for each joint space variable.
- `q_ddot` - The double derivative of the pose. One parameter should be given for each degree of freedom.

2.4 Operational Space XML

Code Sample 4 shows an example operational space XML file. The following can be noted about the structure of this file

1. The document is enclosed by the tag `op_spaces`.
2. Within the `op_spaces` tag there is a tag associated with each set of cables (`op_set`.
3. Within the `op_set` tag there is a tag associated with each operational space `op`.
4. The remaining tags form a description for the operational space.

Listing 4: Example Operational Space XML File.

```
   <?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE cables [
  <!ATTLIST op_set id ID #REQUIRED>
]>

<op_spaces>
  <op_set id="test">
        <position id="1" name="test1">
            <link>2</link>
            <offset>0.0 0.0 1.0</offset>
            <selection_matrix>
                <sx>1</sx>
                <sy>0</sy>
                <sz>0</sz>
            </selection_matrix>
        </position>
    </op_set>
</op_spaces>
```

A more detailed glossary of these tags in addition to all other tags used by the file is provided below.

- `op_spaces` - This tag acts as a container for the set of all operational spaces that describe the mechanism.

- `op_set` - An operational space set represents a set of operational space coordinates. The operational set tag acts as a container for all the operational set information. The tag also contained the attribute `id` which is an identifier for the operational set.
- op - An op level tag is a container for different operational space information. CASPR currently supports the op tags `position`, `pose`, `orientation`. For each op type this will describe the attached link and selection matrix. If position information is used it will also contain the offset in the joint frame.
- `link` - The link that the operational space coordinate is attached to.
- `offset` - The offset of the operational space coordinate relative to the
- `selection_matrix` - The selection matrix within the possible operational space coordinates. This can be used to select only a subset of the possible operational space coordinates.

3 GUI Specifications using XML

XML is also used in CASPR in order to provide the GUI with necessary information regarding which options to load for each box. Written below is a short glossary of each of the GUI XML files which consist of

- Control XML,
- Dynamics XML,
- Kinematics XML,
- Workspace XML.

3.1 Control XML

- `simulator` - This is a container tag to hold all of the xml information.
- `control_class` - This tag currently provides the name of the possible controllers given by the `id` attribute. In future developments the tag will be altered to have properties indicating whether or not ID is required.
- `solver_class` - This tag is associated with a particular inverse dynamics solver. The solver file is given in the `id` attribute. Further details are given in the `solver_type_enum` and `objectives` and `constraints` tags.
- `solver_type_enum` - This tag corresponds to the enum for the desired solver.
- `objectives` - an optional tag that acts as a container for possible objectives.
- `objective` - a tag which represents a particular objective given by the `type` attribute. The tag also contains three tags given by `weight_links_multiplier` (w_{lm}), `weight_cables_multiplier` (w_{cm}) and `weight_constants` (w_c). These indicate the number of unknowns n given by

  $$n = w_{lm} \cdot l + w_{cm} \cdot m + w_c,$$

  where l is the number of joints and m is the number of cables.
- `constraints` - an optional tag that acts as a container for possible constraints.
- `constraint` - a tag which represents a particular constraint given by the `type` attribute. This also uses the tags `weight_links_multiplier`, `weight_cables_multiplier` and `weight_constants`, however the number of elements for these tags changes to reflect the constraint having more than one variable with which to describe it.
- `tuning_parameters` - an optional tag that acts as a container for possible tuning parameters (which may be used when optimisation base methods are not being applied).
- `tuning_parameter` - a tag which plays the same role for `tuning_parameters` as `objective` does for `objectives`.
- `plot_functions` - A container tag to contain all possible plotting options.
- `plot_function` - A tag to represent a function for plotting. This `type` should correspond to a pre-existing plot function defined in a Simulator class.
- `figure_quantity` - The number of figures that are to be generated by the plotting function.

3.2 Dynamics XML

Since the control XML supports the use of inverse dynamics, it provides all the tags that are used in the dynamics XML. As such the inverse dynamics XML uses all tags mentioned within the control XML glossary with the exception of `control_class`.

3.3 Kinematics XML

- `simulator` - This is a container tag to hold all of the xml information.
- `solver_class` - This tag is associated with a particular forward kinematics solver. The solver file is given in the `id` attribute. Further details are given in the `approximation_type_enum` and `q_dot_type_enum`.
- `approximation_type_enum` - This tag corresponds to the enum for the desired approximation method.
- `q_dot_type_enum` - This tag corresponds to the enum for the desired approximation method for computing the numerical derivative of $\mathbf{q}$.
- `plot_functions` - A container tag to contain all possible plotting options.
- `plot_function` - A tag to represent a function for plotting. This `type` should correspond to a pre-existing plot function defined in a Simulator class.
- `figure_quantity` - The number of figures that are to be generated by the plotting function.

3.4 Workspace XML

- `simulator` - This is a container tag to hold all of the xml information.
- `workspace_condition` - This tag is associated with a particular workspace condition given by `id`. It contains the tag `generation_method_enum`.
- `generation_method_enum` - This tag points to the file location for the condition enum.
- `workspace_methods` - A container tag for different workspace metrics
- `workspace_condition` - This tag is associated with a particular workspace metric.
- `grid_types` - This tag is a container for grid types.
- `grid_type` - A tag for an individual type of grid. At this point in time only `UniformGrid` is supported.
- `plot_functions` - A container tag to contain all possible plotting options.
- `plot_function` - A tag to represent a function for plotting. This `type` should correspond to a pre-existing plot function defined in a Simulator class.
- `figure_quantity` - The number of figures that are to be generated by the plotting function.