1. General notions

Let the dynamics of the system be given by the following system of second order differential equations:

, (1.1)

where  is a vector of generalized coordinates.

For a mechanical system this form can be obtained from Lagrange eq. or manipulator eq. It is important that the system is control affine.  is a *joint space inertia matrix* (also known as a *generalized inertia matrix*).

From (1.1) we can find linearized (affine) version of the dynamics. For that we introduce state-space variable . Then the linearized dynamics takes the following form:

, (1.2)

where  is an identity matrix of appropriate dimensions,  is a matrix with all zero elements, ,  (this notation for ,  is only valid for this chapter).

|  |
| --- |
| To access matrix  use function:  g\_dynamics\_Linearization\_SSIM  Symbolic expression is stored in:  SymbolicEngine.LinearizedDynamics.SSIM |

Eq. (1.2) can be re-expressed as follows:

, (1.3)

From this form we can move to a state space representation:

, (1.4)

where , , 

In SRD values ,  and  are not calculated symbolically. Instead values  ,  and  are found:

,

,

.

|  |
| --- |
| To access  ,  and  use the following functions:  g\_dynamics\_Linearization\_RHS\_A()  g\_dynamics\_Linearization\_RHS\_B()  g\_dynamics\_Linearization\_RHS\_c()  Symbolic expressions are stored in:  SymbolicEngine.LinearizedDynamics.RHS\_A  SymbolicEngine.LinearizedDynamics.RHS\_B  SymbolicEngine.LinearizedDynamics.RHS\_c |

These are then used to find ,  and  numerically, whenever the need arises:

,

,

.

This allows to avoid symbolic matrix inversion, which is inefficient and produce overly complicated symbolic expressions.

2. PD controllers

SRD supports three types of PD controllers: normal *PD*, *PD with varying gains* and *computed toque controller*.

Normal **PD controller** is given by the following formula:

, (2.1)

where  and  are *gain matrices*,  and  are desired values of *generalized coordinates*  and *generalized velocities* .

**Computed toque controller** is derived from the assumption that the *control error* dynamics should have the following form:

, (2.2)

where  is control error.

Using the fact that , from which follows that , we can rewrite (2.2) as:



which allows us to express *control actions* :

 (2.3)

This approach only valid for the case when  is invertible (i.e. the system is *fully actuated*).

|  |
| --- |
| SRD provides access to  value via function:  g\_control\_ForcesForComputedTorqueController() |

**PD with varying gains** is a simplified version of the computed toque controller:

. (2.4)

3. Linear quadratic regulators (LQR)

SRD supports iterative linear quadratic regulator - an LQR which gain matrix is updates with a specific update rate.

LQR has the following form:

, (3.1)

where  is the optimal gain matrix,  and  are desired values for  and . Value  is found using the *inverse dynamics* method:

, (3.2)

where  is a pseudo inverse of .

Gain matrix  is found using standard procedure (solving algebraic Riccati eq.) for linearization around point , .

4. Model predictive controller (MPC)

To formulate the MPC problem we rewrite the dynamics of the system in a discrete form:

, (4.1)

where ,  and  are values of ,  and  obtained by linearizing around point .

We will consider  *prediction steps*, with initial state of the system being . The first step is to construct a system of linear equations that arise from the system’s dynamics. Expression (4.1) can be rewritten as:

. (4.2)

We introduce the following notation:

,  (4.3)

Using this notation, we can rewrite the system (4.2) in the following way:

, (4.4)

where:

.

, .

.





We introduce new variables: , , , which allow us to rewrite (4.4) as:

. (4.5)

Then the dynamics over  prediction steps has form:

. (4.6)

We need to also take into account initial conditions: . This means that system (4.6) gets another row:

. (4.7)

where ,  has the same number of columns as . Here  is the known actual state the system occupies at the moment.

Finally, the optimization problem for this MPC is given as follows:

**minimize** 

**subject to** .

where  is a weight matrix for quadratic cost and  is a weight vector for linear cost. The optimization problem is QP (quadratic programming) problem with linear equality constraints and quadratic cost.

The value  can be found as , and  can be then extracted from  .