Dynamic model in task space coordinates

This live script obtains a closed formula for the inverse dynamics of the Otbot by using a tricky elimination of the Lagrange multipliers. It then obtains the equation of motion in task-space coordinates, which can be used to design a computed-torque controller for trajectory tracking.

Table of Contents

[1 - Initializations](#MW_H_9D15A764)   
[2 - Multiplier-free inverse dynamics](#MW_H_5145A916)   
[3 - Equation of motion in task-space coordinates](#MW_H_002CD5AD)   
[4 - Save main matrices](#MW_H_184800FE)   
[5 - Print ellapsed time](#MW_H_A30E5B9A)

# 1 - Initializations

% Clear all variables, close all figures, and clear the command window

clearvars

close all

clc

% Start stopwatch

tic;

% Display the matrices with rectangular brackets

sympref('MatrixWithSquareBrackets',true);

% Avoid Matlab's own substitution of long expressions

sympref('AbbreviateOutput',false);

% Symbolic variables to be used (see the figures and explanations below)

syms x y % Absolute coords of the pivot joint

syms x\_dot y\_dot % Absolute velocity components of the pivot joint

syms alpha % Absolute angle of the platform

syms alpha\_dot % Absolute angular velocity of the platform

syms varphi\_r % Angle of right wheel

syms varphi\_l % Angle of left wheel

syms varphi\_p % Pivot joint angle

syms varphi\_dot\_l % Angular velocity of left wheel

syms varphi\_dot\_r % Angular velocity of right wheel

syms varphi\_dot\_p % Angular velocity of the pivot motor

syms l\_1 % Pivot offset relative to the wheels axis

syms l\_2 % One half of the wheels separation

syms m\_c % Mass of the chassis including the wheels

syms m\_p % Mass of the platform

syms x\_B y\_B % Coords of the c.o.m. of the chassis in chassis frame

syms x\_F y\_F % Coords of the c.o.m. of the platform in platform frame

syms I\_c % Vertical moment of inertia of the chassis at B

syms I\_p % Vertical moment of inertia of the platform at F

syms I\_a % Axial moment of inertia of one wheel

% Load matrices

load('MIIK.mat')

load('MFIK.mat')

load('M.mat')

load('C.mat')

# 2 - Multiplier-free inverse dynamics

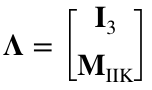
Our goal is to obtain an equation of motion that describes the time evolution of the  coordinates alone, and at the same time does not contain the annoying Lagrange multipliers . In this way we will obtain a one-to-one relationship between platform accelerations and the torques  applied to the robot. This relationship can later be used to design a computed-torque control law.

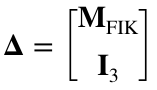
Consider the following parametrizations of the feasible 

,

,

where





Lambda = [eye(3); MIIK]

Delta = [MFIK; eye(3)]

For later use, also consider the time derivative of the first parameterization

.

and let us compute  with Matlab:

% Substitute its variables by functions of t

syms f1(t) f2(t)

Lambda2 = subs(Lambda,[alpha varphi\_p],[f1(t) f2(t)]);

% Take the time derivative

dLambdadt = diff(Lambda2,t);

% Substitute d/dt of f1 and f2 by the original variables using dot notation

Lambda\_dot = subs(dLambdadt,...

[f1(t) f2(t) diff(f1,t) diff(f2,t)], ...

[alpha varphi\_p alpha\_dot varphi\_dot\_p])

clearvars f1(t) f2(t)

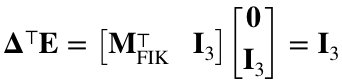
Recall that the equation of motion of the Otbot takes the form



Let us multiply this equation by :



Note that , as the columns of  form a basis of the kernel of , and  is a vector orthogonal to this kernel. Also note that

,

so the equation of motion reduces to



This formula gives us an explicit solution for the inverse dynamics of the Otbot.

# 3 - Equation of motion in task-space coordinates

If we now substitute  and  into the earlier equation we obtain:



Let us compute





M\_bar = simplify(Delta.' \* M \* Lambda)

C\_bar = simplify(Delta.' \* (M \* Lambda\_dot + C \* Lambda) )

We call these matrices the task-space mass matrix, and the task-space Coriolis matrix respectively. Using them, the equation of motion can be written in the compact form



This is called the equation of motion in task-space coordinates, and can be used to design a computed-torque controller to track arbitrary trajectories in task space.

# 4 - Save main matrices

save('Lambda.mat',"Lambda")

save('Lambda\_dot.mat',"Lambda\_dot")

save('Delta.mat',"Lambda")

save('M\_bar.mat',"M\_bar")

save('C\_bar.mat',"C\_bar")

# 5 - Print ellapsed time

toc