Modeling and control of the rodwheel

Luc Jaulin

Lab-Sticc, ENSTA-Bretagne

Abstract. The rodwheel is a wheel equipped with a rod motorized on the axle. This paper proposes a Lagrangian approach to find the state equations of the rodwheel rolling on a plane without friction. The approach takes advantage of a symbolic computation. A controller is proposed to stabilize the rodwheel with the rod upward and going straight at a desired speed.

1 Introduction

Consider a disk rolling on a plane without friction with a rod which can move along the wheel plane [4] as shown on Figure 1. In the axle of the wheel, we have a motor which produces a torque u between the wheel and the rod.

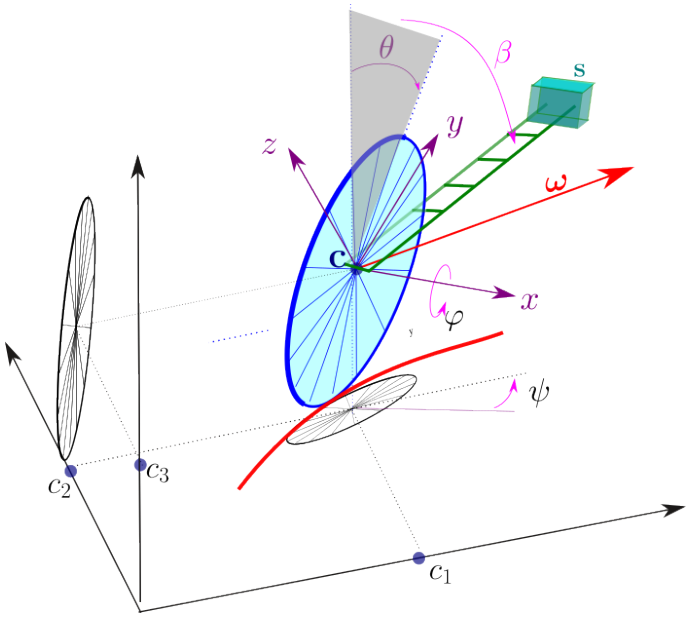

Fig. 1: Disk rolling on a plane with a rod

The disk has a mass $m = 5$kg and its radius is $r = 1m$. The gravity is taken as $g = 9.81ms^{-2}$. The mass $\mu = 1$kg of the rod is reduced to the single point $\mathbf{s}$ which is at a distance $\ell = 2m$ to the center $\mathbf{c}$ of the wheel.

In this paper, we want to find the state equations describing the motion of the rodwheel. The rolling wheel model has already been found since a long time (see *e.g.* [1]) and even highly studied since (see *e.g.* [8]). Extension to more general wheel based vehicle have also been proposed (see *e.g.* [2] for the bicycle). The problem of the wheel with the rod seems to be original and I believe it can be used as a fast, light terrestrial robot to move in a urban environment with a reduced amount of batteries. Computing the state equations for a rodwheel is a tedious task. In this paper, I want to take advantage of symbolic computing (here the `sympy`

package of `Python`) in order to derive these state equations. The Lagrangian approach [9], often applied to model robots [3], will be chosen.

Once we have found the state model of the rodwheel, we will propose a controller to make the rodwheel going straight ahead at a desired speed with the rod upward. The control approach uses techniques dedicated to underactuated mechanical systems [5].

2 Modeling

2.1 State vector

We take the state vector $\mathbf{x} = (c_1, c_2, \varphi, \theta, \psi, \beta, \dot{\varphi}, \dot{\theta}, \dot{\psi}, \dot{\beta})$ where (c_1, c_2) is the vertical projection of center $\mathbf{c} = (c_1, c_2, c_3)$ of the disk, φ, θ, ψ are the three Euler angles and β is the rod angle. As illustrated by Figure 1,

- φ is the spin angle
- θ is the stand angle, *i.e.*, when $\theta = 0$, the disk is vertical
- ψ is the heading, *i.e.*, the horizontal orientation of the disk
- β is the angle of the rod with respect to the vertical axis.

Therefore, we should be able to find a state equations of the form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, u). \tag{1}$$

to represent the dynamics of the rodwheel. For this, we will follow the approach presented in [7].

To use `sympy`, we first declare the symbolic variables and functions to be used:

```
from sympy import *
t=symbols('t')
m,g,r,μ,l= symbols('m g r, μ,l')
c1,c2 = Function('c1')('t'),Function('c2')('t')
dc1,dc2 = Function('dc1')('t'),Function('dc2')('t')
ddc1,ddc2 = Function('ddc1')('t'),Function('ddc2')('t'),
φ,θ,ψ,β = Function('φ')('t'),Function('θ')('t'),Function('ψ')('t'),Function('β')('t')
dφ,dθ,dψ,dβ = Function('dφ')('t'),Function('dθ')('t'),
              Function('dψ')('t'),Function('dβ')('t')
ddφ,ddθ,ddψ,ddβ = Function('ddφ')('t'),Function('ddθ')('t'),
                  Function('ddψ')('t'),Function('ddβ')('t')
λ1,λ2 = Function('λ1')('t'),Function('λ2')('t')
```

2.2 Orientation

The orientation of the disk is fixed by the three Euler angles φ, θ, ψ. The corresponding orientation matrix is

$$\mathbf{R}_{\text{euler}}(\varphi, \theta, \psi) = \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & -\sin\varphi \\ 0 & \sin\varphi & \cos\varphi \end{pmatrix} \tag{2}$$

It is built by the following `sympy` function

```
def Reuler(φ,θ,ψ):
   Rφ = Matrix([[1,0,0],[0,cos(φ),-sin(φ)],[0,sin(φ),cos(φ)]])
   Rθ = Matrix([[cos(θ),0,sin(θ)],[0,1,0],[-sin(θ),0,cos(θ)]])
   Rψ = Matrix([[cos(ψ),-sin(ψ),0],[sin(ψ),cos(ψ),0],[0,0,1]])
   Return  Rψ*Rθ*Rφ
```

The rotation vector depends on the Euler angles and their derivatives. Its expression [6] can be obtained using the relation

$$\mathbf{R}^{\mathrm{T}}\dot{\mathbf{R}} = \begin{pmatrix} 0 & -\omega_{r3} & \omega_{r2} \\ \omega_{r3} & 0 & -\omega_{r1} \\ -\omega_{r2} & \omega_{r1} & 0 \end{pmatrix} \tag{3}$$

which gives us the following `sympy` function

```
def wr(R):
   W=Transpose(R)*diff(R,t)
   return Matrix([[-W[1,2]],[W[0,2]],[-W[0,1]]])
```

2.3 Lagrangian

In order, to get the state equations the rodwheel, we use a Lagrangian approach. For this, we need to express the Lagrangian $\mathcal{L}$ with respect to the state variables. Recall that

$$\mathcal{L} = E_K - E_p \tag{4}$$

where E_K is the kinetic energy and E_p is the potential energy. We have

$$E_K = \frac{1}{2}\boldsymbol{\omega}_r^{\mathrm{T}}\mathbf{I}\boldsymbol{\omega}_r + \frac{1}{2}m\|\dot{\mathbf{c}}\|^2 + \frac{1}{2}\mu\|\dot{\mathbf{s}}\|^2 \tag{5}$$

where $\mathbf{c}$ is the center of the wheel, $\mathbf{s}$ is the end point of the rod given by

$$\mathbf{s} = \mathbf{c} + \mathbf{R}_{\mathrm{euler}}(\beta, \theta, \psi)$$

and

$$\mathbf{I} = \begin{pmatrix} \frac{mr^2}{2} & 0 & 0 \\ 0 & \frac{mr^2}{4} & 0 \\ 0 & 0 & \frac{mr^2}{4} \end{pmatrix} \tag{6}$$

is the inertia matrix of the disk. Denote by

$$\mathbf{q} = (c_1, c_2, \varphi, \theta, \psi, \beta) \tag{7}$$

the generalized coordinates of the system, *i.e.*, the degrees of freedom. The Lagrangian, which is a function of $(\mathbf{q}, \dot{\mathbf{q}})$, is computed with `sympy` as follows:

```
def Lagrangian(q,dq):
   c1,c2,φ,θ,ψ,β=list(q)
   R = Reuler(φ,θ,ψ)
   c=Matrix([[c1],[c2],[r*cos(θ)]])
   dc=diff(c,t)
   s=Reuler(β,θ,ψ)*Matrix([[0],[0],[1]])+c
   ds=diff(s,t)
   Ep=m*g*c3+μ*s[2]
   I=Matrix([[1/2*m*r**2,0,0],[0,1/4*m*r**2,0],[0,0,1/4*m*r**2]])
   Ek=1/2*m*(dc.dot(dc))+1/2*μ*(ds.dot(ds))+1/2*wr(R).dot(I*wr(R))
   L=subsdiff(Ek-Ep)
   return L
```

where `subsdiff(E)` transforms the differentiation operator into the corresponding state variables:

```
def subsdiff(E):
   E=E.subs({diff(dc1,t): ddc1,diff(dc2,t): ddc2,  diff(dφ,t): ddφ,
             diff(dθ,t): ddθ, diff(dψ,t): ddψ, diff(dβ,t): ddβ})
   E=E.subs({diff(c1,t): dc1,diff(c2,t): dc2,  diff(φ,t): dφ,
             diff(θ,t): dθ,diff(ψ,t): dψ, diff(β,t): dβ})
   return simplify(E)
```

We get

$$
\begin{aligned}
\mathcal{L}(\mathbf{q},\dot{\mathbf{q}}) \quad = \quad & \tfrac{m}{8}r^2(\dot\theta\sin\varphi-\dot\psi\cos\theta\cos\varphi)^2+\tfrac{m}{8}r^2(\dot\theta\cos\varphi+\dot\psi\sin\varphi\cos\theta)^2 \\
& +\tfrac{m}{4}r^2(\dot\varphi-\dot\psi\sin\theta)^2+\tfrac{m}{2}(r^2\dot\theta^2\sin^2\theta+\dot c_1^2+\dot c_2^2) \\
& -\mu(\ell\cos\beta+r)\cos\theta \\
& +\tfrac{\mu}{2}\left(\ell\sin\psi\cos\beta\cos\theta\dot\theta+\ell\left(\sin\beta\sin\psi+\sin\theta\cos\beta\cos\psi\right)\dot\psi-\ell(\sin\beta\sin\theta\sin\psi+\cos\beta\cos\psi)\dot\beta+\dot c_2\right)^2 \\
& +\tfrac{\mu}{2}\left(\ell\cos\beta\cos\theta\cos\psi\dot\theta+\ell\left(\sin\beta\cos\psi-\sin\theta\sin\psi\cos\beta\right)\dot\psi+\ell\left(\sin\psi\cos\beta-\sin\beta\sin\theta\cos\psi\right)\dot\beta+\dot c_1\right)^2 \\
& +\tfrac{\mu}{2}\left(\ell\sin\beta\cos\theta\dot\beta+(\ell\sin\theta\cos\beta+r\sin\theta)\dot\theta\right)^2 \\
& -gmr\cos\theta
\end{aligned}
$$

In what follows, we will avoid writing such complex expressions (which can be far larger than this one). Indeed, these expressions should remain transparent for the user since is it handled by `sympy`.

2.4 Euler-Lagrange equations

The evolution of $\mathbf{q}$ obeys to the Euler-Lagrange equation for non-holonomic systems:

$$
\underbrace{\frac{d}{dt}\left(\frac{\partial\mathcal{L}}{\partial\dot{\mathbf{q}}}\right)-\frac{\partial\mathcal{L}}{\partial\mathbf{q}}}_{\mathcal{Q}(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}})}=\underbrace{\begin{pmatrix}\tau_{\mathbf{c}_1}\\ \tau_{\mathbf{c}_2}\\ \tau_\varphi\\ \tau_\theta\\ \tau_\psi\\ \tau_\beta\end{pmatrix}}_{\boldsymbol{\tau}}+\begin{pmatrix}0\\0\\1\\0\\0\\-1\end{pmatrix}\cdot u \tag{8}
$$

where the right hand side corresponds the generalized forces. It is composed of the forces $\boldsymbol{\tau}$ generated by the ground and of the motor forces corresponding to internal torque. The entries for the motor force vector are consistent with the fact that the motor produces the torque u between the degrees of freedom β (the rod) and φ (the wheel).

Remark 1. Note that $\mathcal{Q}(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}})$ is affine in $\ddot{\mathbf{q}}$. This property will be used later to build the state space model of the rodwheel.

An expression for $\mathcal{Q}(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}})$ is obtained using `sympy` by:

```
q=Matrix([c1,c2,φ,θ,ψ,β])
dq=Matrix([dc1,dc2,dφ,dθ,dψ,dβ])
ddq=Matrix([ddc1,ddc2,ddφ,ddθ,ddψ,ddβ])
L=Lagrangian(q,dq)
Q=subsdiff(diff(L.jacobian(dq),t)-L.jacobian(q))
```

The generalized forces $\tau_{\mathbf{c}_1},\tau_{\mathbf{c}_2},\tau_\varphi,\tau_\theta,\tau_\psi,\tau_\beta$ generated by the reaction of the ground are linked by constraints that should now be written.

2.5 Non holonomic constraints

Without any reaction constraint of the ground onto the disk, the state vector would have been

$$(\mathbf{q}, \dot{\mathbf{q}}) = (c_1, c_2, \varphi, \theta, \psi, \beta, \dot{c}_1, \dot{c}_2, \dot{\varphi}, \dot{\theta}, \dot{\psi}, \dot{\beta}), \tag{9}$$

i.e., it would have been composed of the degrees of freedom $\mathbf{q}$ and they derivatives $\dot{\mathbf{q}}$. Now, due to the ground forces, the variables ($\mathbf{q}$,$\dot{\mathbf{q}}$) are linked by some differential constraints. These constraints will be needed to derive the state equations with $\mathbf{x} = (c_1, c_2, \varphi, \theta, \psi, \beta, \dot{\varphi}, \dot{\theta}, \dot{\psi}, \dot{\beta})$ as a state vector. Since we have two variables to eliminate (here $\dot{c}_1$ and $\dot{c}_2$), we need to find two more differential constraints. These two constraints are those generated by the ground forces. They translate the fact that the point of the disk in contact with the ground has a zero velocity. It means that the disk neither slides tangentially (it could be the first equation) nor laterally (it could be the second equation). We understand that these two equations have the form

$$\begin{array}{rcl} \dot{c}_1 & = & \alpha_1 \cdot \dot{\varphi} + \alpha_2 \cdot \dot{\theta} + \alpha_3 \cdot \dot{\psi} \\ \dot{c}_2 & = & \beta_1 \cdot \dot{\varphi} + \beta_2 \cdot \dot{\theta} + \beta_3 \cdot \dot{\psi} \end{array} \tag{10}$$

where the α_i's and the $\beta_i's$ depend on $\mathbf{q}$. More precisely, as shown in [7] these constraints are

$$\begin{array}{rcl} \dot{c}_1 & = & r \sin\psi \cdot \dot{\varphi} + r \cos\psi \cos\theta \cdot \dot{\theta} - r \sin\psi \sin\theta \cdot \dot{\psi} \\ \dot{c}_2 & = & -r \cos\psi \cdot \dot{\varphi} + r \sin\psi \cos\theta \cdot \dot{\theta} + r \cos\psi \sin\theta \cdot \dot{\psi} \end{array} \tag{11}$$

Figure 2 illustrates how these formulas are obtained . The left subfigure shows that when $\dot{\theta} = 0$ and $\dot{\psi} = 0$, we have

$$\begin{array}{rcl} \dot{c}_1 & = & r \sin\psi \cdot \dot{\varphi} \\ \dot{c}_2 & = & -r \cos\psi \cdot \dot{\varphi} \end{array} \tag{12}$$

The subfigure in the center illustrates that if $\dot{\varphi} = 0$ and $\dot{\psi} = 0$,

$$\begin{array}{rcl} \dot{c}_1 & = & r \cos\psi \cos\theta \cdot \dot{\theta} \\ \dot{c}_2 & = & r \sin\psi \cos\theta \cdot \dot{\theta} \end{array} \tag{13}$$

The right subfigure illustrates that if $\dot{\theta} = 0$ and $\dot{\varphi} = 0$,

$$\begin{array}{rcl} \dot{c}_1 & = & -r \sin\psi \sin\theta \cdot \dot{\psi} \\ \dot{c}_2 & = & r \cos\psi \sin\theta \cdot \dot{\psi} \end{array} \tag{14}$$

By superposition, we get Equation 11. These constraints are said to be non holonomic since they will not allow us to express our system with a state composed of some degrees of freedom $\mathbf{q}$ and their derivatives $\dot{\mathbf{q}}$.

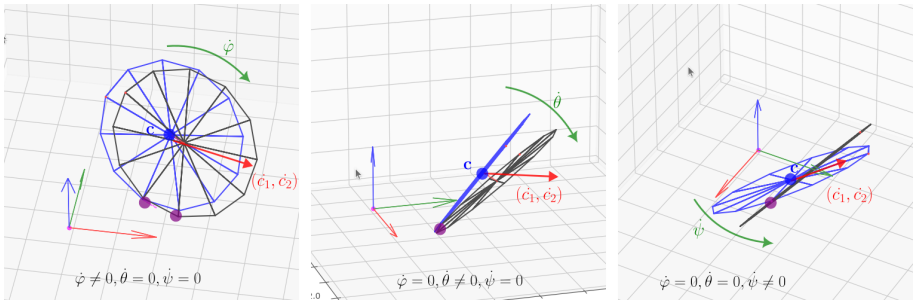

Fig. 2: Wheel rolling on a plane (the rod is not represented)

2.6 D'Alembert's principle

We need to find an expression for $\boldsymbol{\tau}$ which occurs in the right hand side of (8). The components for $\boldsymbol{\tau}$ correspond to the generalized ground forces applied to our system. Now, since the contact point has no velocity, the ground force cannot modify the energy of the system.

In order to use this information, let us to recall the principle of d'Alembert: *for arbitrary virtual displacements, the constraint forces don't do any work.*

The virtual displacements are infinitesimal changes $\delta\mathbf{q} = \mathbf{q}(t+dt) - \mathbf{q}(t)$ that should be consistent with some feasible trajectories $\mathbf{q}(t)$. For our rolling system, the virtual displacements satisfy

$$\begin{array}{rcll} \delta c_1 - r\sin\psi\cdot\delta\varphi - r\cos\psi\cos\theta\cdot\delta\theta + r\sin\psi\sin\theta\cdot\delta\psi & = & 0 & (i) \\ \delta c_2 + r\cos\psi\cdot\delta\varphi - r\sin\psi\cos\theta\cdot\delta\theta - r\cos\psi\sin\theta\cdot\delta\psi & = & 0 & (ii) \end{array} \tag{15}$$

for the same reasons than those used to derive (11). The fact that there is no work translates into

$$\delta W = \boldsymbol{\tau}^{\mathrm{T}}\cdot\delta\mathbf{q} = \tau_{c_1}\cdot\delta c_1 + \tau_{c_2}\cdot\delta c_2 + \tau_\varphi\cdot\delta\varphi + \tau_\theta\cdot\delta\theta + \tau_\psi\cdot\delta\psi \quad + \tau_\beta\cdot\delta\beta = \quad 0. \tag{16}$$

Equivalently, Equation 16 is a linear combination of the two equations (15), *i.e.*, (16) = $\lambda_1\cdot$(15,i)+$\lambda_2\cdot$(15,ii). The λ_i's are called the *Lagrange parameters*. Therefore:

$$\begin{pmatrix} \tau_{c_1} \\ \tau_{c_2} \\ \tau_\varphi \\ \tau_\theta \\ \tau_\psi \\ \tau_\beta \end{pmatrix} = \lambda_1\cdot\begin{pmatrix} 1 \\ 0 \\ -r\sin\psi \\ -r\cos\psi\cos\theta \\ r\sin\psi\sin\theta \\ 0 \end{pmatrix} + \lambda_2\cdot\begin{pmatrix} 0 \\ 1 \\ r\cos\psi \\ -r\sin\psi\cos\theta \\ -r\cos\psi\sin\theta \\ 0 \end{pmatrix} \tag{17}$$

or in a vector form:

$$\boldsymbol{\tau} = \mathbf{A}^{\mathrm{T}}(\mathbf{q})\begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix} \tag{18}$$

where

$$\mathbf{A}(\mathbf{q}) = \begin{pmatrix} 1 & 0 & -r\sin\psi & -r\cos\psi\cos\theta & r\sin\psi\sin\theta & 0 \\ 0 & 1 & r\cos\psi & -r\sin\psi\cos\theta & -r\cos\psi\sin\theta & 0 \end{pmatrix}. \tag{19}$$

2.7 State equations

Using (8) with (18), we get

$$\mathcal{Q}(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}}) - \boldsymbol{\tau}(\mathbf{q},\boldsymbol{\lambda}) = \begin{pmatrix} 0 \\ 0 \\ u \\ 0 \\ 0 \\ -u \end{pmatrix} \tag{20}$$

This system is made of 6 equations which are linear in 8 variables : $\boldsymbol{\lambda} = (\lambda_1, \lambda_2)$ and $\ddot{\mathbf{q}} = (\ddot{c}_1, \ddot{c}_2, \ddot{\varphi}, \ddot{\theta}, \ddot{\psi}, \ddot{\beta})$ (see Equation 18 and Remark 1). In order to square the system, we need to add two equations (to get 8 equations). They are obtained by differentiating of the non-holonomic constraints (11) given by

$$\underbrace{\mathbf{A}(\mathbf{q})\cdot\dot{\mathbf{q}}}_{\mathbf{a}(\mathbf{q},\dot{\mathbf{q}})} = \mathbf{0}. \tag{21}$$

Let us differentiate this equation. We get:

$$\underbrace{\frac{\partial\mathbf{a}(\mathbf{q},\dot{\mathbf{q}})}{\partial\mathbf{q}}\cdot\dot{\mathbf{q}} + \frac{\partial\mathbf{a}(\mathbf{q},\dot{\mathbf{q}})}{\partial\dot{\mathbf{q}}}\cdot\ddot{\mathbf{q}}}_{=\frac{d}{dt}\mathbf{a}(\mathbf{q},\dot{\mathbf{q}})} = \mathbf{0} \tag{22}$$

Adding these two equations to (20) yields

$$\begin{cases} \frac{d}{dt}\mathbf{a}(\mathbf{q},\dot{\mathbf{q}}) & = & \begin{pmatrix} 0 \\ 0 \end{pmatrix} \\ \mathcal{Q}(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}}) - \boldsymbol{\tau}(\mathbf{q},\boldsymbol{\lambda}) & = & \begin{pmatrix} 0 \\ 0 \\ u \\ 0 \\ 0 \\ -u \end{pmatrix} \end{cases}$$

or equivalently

$$\underbrace{\begin{pmatrix} \frac{\partial \mathbf{a}(\mathbf{q},\dot{\mathbf{q}})}{\partial \mathbf{q}} \cdot \dot{\mathbf{q}} + \frac{\partial \mathbf{a}(\mathbf{q},\dot{\mathbf{q}})}{\partial \dot{\mathbf{q}}} \cdot \ddot{\mathbf{q}} \\ \mathcal{Q}(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}}) - \boldsymbol{\tau}(\mathbf{q},\boldsymbol{\lambda}) \end{pmatrix}}_{\mathcal{S}(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}},\boldsymbol{\lambda})} = \underbrace{\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ -1 \end{pmatrix}}_{\mathbf{b}_u} \cdot u \tag{23}$$

Since $\mathcal{S}(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}},\boldsymbol{\lambda})$ is affine in $\boldsymbol{\lambda},\ddot{\mathbf{q}}$ (see Equation 18 and Remark 1), we have

$$\mathcal{S}(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}},\boldsymbol{\lambda}) = \underbrace{\begin{pmatrix} \frac{\partial}{\partial \boldsymbol{\lambda}}\mathcal{S}(\mathbf{q},\dot{\mathbf{q}},\mathbf{0},\mathbf{0}) & \frac{\partial}{\partial \ddot{\mathbf{q}}}\mathcal{S}(\mathbf{q},\dot{\mathbf{q}},\mathbf{0},\mathbf{0}) \end{pmatrix}}_{=\mathbf{M}(\mathbf{q})} \cdot \begin{pmatrix} \boldsymbol{\lambda} \\ \ddot{\mathbf{q}} \end{pmatrix} + \underbrace{\mathcal{S}(\mathbf{q},\dot{\mathbf{q}},\mathbf{0},\mathbf{0})}_{=-\mathbf{b}(\mathbf{q},\dot{\mathbf{q}})}. \tag{24}$$

where $\mathbf{M}(\mathbf{q})$ is a square matrix called the *mass matrix* (we can show that it does not depend on $\dot{\mathbf{q}}$, this is why we have written $\mathbf{M}(\mathbf{q})$ instead of $\mathbf{M}(\mathbf{q},\dot{\mathbf{q}})$). Thus (23) becomes

$$\mathbf{M}(\mathbf{q}) \cdot \begin{pmatrix} \boldsymbol{\lambda} \\ \ddot{\mathbf{q}} \end{pmatrix} = \mathbf{b}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{b}_u \cdot u. \tag{25}$$

Isolating $\ddot{\mathbf{q}}$, we get

$$\begin{pmatrix} \ddot{\varphi} \\ \ddot{\theta} \\ \ddot{\psi} \\ \ddot{\beta} \end{pmatrix} = \underbrace{\begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}}_{\mathbf{P}} \cdot \mathbf{M}^{-1}(\mathbf{q}) \cdot (\mathbf{b}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{b}_u \cdot u) \tag{26}$$

where $\mathbf{P}$ is the projection matrix which selects the four last components of the vector $(\lambda_1,\lambda_2,\ddot{c}_1,\ddot{c}_2,\ddot{\varphi},\ddot{\theta},\ddot{\psi},\ddot{\beta})$.

The expression for $\mathbf{M}$ and $\mathbf{b}$ are obtained by:

```
A=Matrix([[1,0,-r*sin(ψ),-r*cos(ψ)*cos(θ),r*sin(ψ)*sin(θ),0],
          [0,1, r*cos(ψ),-r*sin(ψ)*cos(θ),-r*cos(ψ)*sin(θ),0]])
τ=λ1*A[0,:]+λ2*A[1,:]
a=A*dq
da=diff(a,t)
S=Matrix([da,*list(Q-τ)])
S=subsdiff(S)
M=S.jacobian([λ1,λ2,ddq])
b=-S.subs({λ1:0,λ2:0,ddc1:0,ddc2:0,ddφ:0, ddθ:0, ddψ:0, ddβ:0})
```

Therefore, the state equations of the rodwheel are

$$\begin{array}{rcll}
\begin{pmatrix} \dot{c}_1 \\ \dot{c}_2 \end{pmatrix} & = & r \begin{pmatrix} \sin\psi & \cos\psi\cos\theta & -\sin\psi\sin\theta \\ -\cos\psi & \sin\psi\cos\theta & \cos\psi\sin\theta \end{pmatrix} \begin{pmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} & \text{(see (11))} \\
\begin{pmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \\ \dot{\beta} \end{pmatrix} & = & \begin{pmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \\ \dot{\beta} \end{pmatrix} & \\
\begin{pmatrix} \ddot{\varphi} \\ \ddot{\theta} \\ \ddot{\psi} \\ \ddot{\beta} \end{pmatrix} & = & \mathbf{P}\cdot\mathbf{M}^{-1}(\mathbf{q})\cdot(\mathbf{b}(\mathbf{q},\dot{\mathbf{q}})+\mathbf{b}_u\cdot u) & \text{(see (26))}
\end{array} \tag{27}$$

The symbolic computation is only used to generate the `Python` functions for $\mathbf{M}(\mathbf{q})$ and $\mathbf{b}(\mathbf{q},\dot{\mathbf{q}})$. Getting an expression for $\mathbf{v}(\mathbf{q},\dot{\mathbf{q}},u)=\mathbf{M}^{-1}(\mathbf{q})\cdot(\mathbf{b}(\mathbf{q},\dot{\mathbf{q}})+\mathbf{b}_u\cdot u)$ would have been too heavy for `sympy`. This is why the choice has been made to compute $\mathbf{v}(\mathbf{q},\dot{\mathbf{q}},u)$ numerically during the simulation by solving the following linear system

$$\mathbf{M}(\mathbf{q})\cdot\mathbf{v}=\mathbf{b}(\mathbf{q},\dot{\mathbf{q}})+\mathbf{b}_u\cdot u$$

each time we want to evaluate the evolution function $\mathbf{f}(\mathbf{x},u)$. To build the `Python` function from a `sympy` expression, we write

```
M,b=lambdify((c1,c2,φ,θ,ψ,β,dφ,dθ,dψ,dβ,m,g,r,μ,l),(M,b))
```

3 Control

We want to build a controller for the rodwheel and validate on a simulation. We first have to build a simulator based on the state space model.

3.1 Simulation

The `Python` code for the evolution function is:

```
def f(x,u):
   c1,c2,φ,θ,ψ,β,dφ,dθ,dψ,dβ=list(x.flatten())
   dc1= r*sin(ψ)*dφ+r*cos(ψ)*cos(θ)*dθ-r*sin(θ)*sin(ψ)*dψ
   dc2=-r*cos(ψ)*dφ+r*sin(ψ)*cos(θ)*dθ+r*sin(θ)*cos(ψ)*dψ
   bu=array([[0],[0],[0],[0],[1],[0],[0],[-1]])
   v=np.linalg.solve(M,b+bu*u)
   ddφ,ddθ,ddψ,ddβ=list((v[4:8]).flatten())
   return array([[dc1],[dc2],[dφ],[dθ],[dψ],[dβ],[ddφ],[ddθ],[ddψ],[ddβ]])
```

We use a Runge-Kutta integration scheme for the simulation

```
x=x+dt*(0.25*f(x,u)+0.75*(f(x+(2/3)*dt*f(x,u),u)))
```

We chose the initial state vector as

$$\left(c_1,c_2,\varphi,\theta,\psi,\dot{\varphi},\dot{\theta},\dot{\psi}\right)=(4,0,0,0.3,0,-0.5,6,-3,0,0)\,. \tag{28}$$

and the parameters as

$$(m,g,r,\mu,\ell)=(5,9.81,1,1,2). \tag{29}$$

The result of the simulation without any control (*i.e.*, for $u = 0$) is represented on Figure 3. The rod is painted green. The blue wheel corresponds to $t = 0$ and the red wheel to $t = 8$. The evolution of the wheel's center and of the contact point are represented in red and magenta respectively. The simulation highlights the chaotic behavior of the rodwheel.

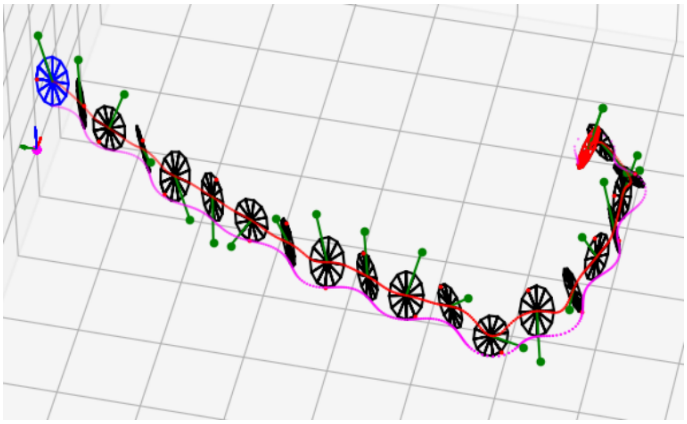

Fig. 3: Evolution of rodwheel with no control

3.2 Controller

We want a controller that makes our rodwheel going straight, the rod upward at a given speed. Several difficulties have to be raised:

- We have a single input u for several state variables to control.
- The equations of the system are complex. It is difficult to write a closed form for the state equations in less than one page.
- The system is not controllable once the rodwheel goes straight.
- The system is unstable with respect to the rod angle β.
- The system is unstable with respect to the stand angle θ.

Concerning the proof of the non controllability, instead of using the model, it is more direct to use the symmetries of the problem. Assume that $\dot{\psi} = 0$, $\theta = 0, \dot{\theta} = 0$ (this is what we have once our goal is reached). We have an evolution which remains symmetric with respect to the vertical plane supported by the wheel. Thus, for any control u, we are trapped in a situation where $\dot{\psi} = 0$, $\theta = 0, \dot{\theta} = 0$.

For the instability a linearisation around $\dot{\psi} = 0$, $\theta = 0, \dot{\theta} = 0$ could be used. Now, the instability with respect to β is intuitive and the instability with respect to θ is observed by simulation (see further). Indeed, when the rod is stabilized above **c**, the precession tends to amplify.

In what follows, we propose a very simple expression for a possible controller. This will be enough to illustrate that the robot can be controlled with u as a single input.

Case 1. Consider the situation where the initial stand angle θ and its derivative $\dot{\theta}$ are both equal to zero. Due to the symmetry of the system, $\theta(t) = 0$ for all t. Take the following proportional and derivative controller

$$\begin{array}{rcl} u & = & 20(\beta - \beta_0) + 20\dot{\beta} \\ \beta_0 & = & \tanh(2 - \dot{\varphi}) \end{array} \tag{30}$$

where $\beta - \beta_0$ is the error. If the closed loop system is stable, then $\beta \to \beta_0$ (this notations means that β converges to β_0 when $t \to \infty$). Now, the robot will accelerate (or decelerate) until $\beta_0 = 0$, *i.e.*, $\dot{\varphi} = 2$. This means that

we can control the speed of the rodwheel, at least when the stand angle $\theta(t)$ is equal to zero. Figure 4 shows the behavior of the closed loop for the initial state :

$$(c_1, c_2, \varphi, \theta, \psi, \beta, \dot{\varphi}, \dot{\theta}, \dot{\psi}, \dot{\beta}) = (4, 0, 0, 0, 0, \pi, 0, 0, 0, 0) \tag{31}$$

For the initialization, the rod is downward and stands up in order to create the right acceleration. We observe that $\dot{\varphi} \to 2$ and that $\beta \to 0$.

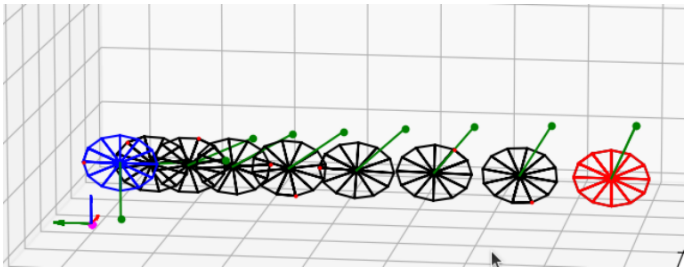

Fig. 4: Rodwheel with $\theta(t) = 0$ which accelerates to reach the speed of $\dot{\varphi} = 2$

To show the instability with respect to θ, take $\theta(0) = 2 \cdot 10^{-12}$ instead of $\theta(0) = 0$. The simulation generates Figure 5. We see the rodwheel falling strongly after few seconds.

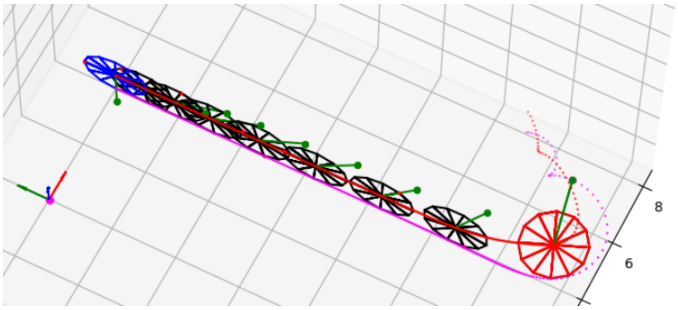

Fig. 5: A tiny perturbation with respect to θ generates a fall of the robot

To control the stand angle, we propose to add a term which accelerates when θ deviates from zero. We get the following controller:

$$\begin{array}{rcl} u & = & 5(\beta - \beta_0) + 5\dot{\beta} + 20 \cdot |\theta| \\ \beta_0 & = & 0.2 \cdot \tanh(10 - \dot{\varphi}) \end{array} \tag{32}$$

The initial vector is the same as in Subsection 3.1, *i.e.*,

$$\left(c_1, c_2, \varphi, \theta, \psi, \dot{\varphi}, \dot{\theta}, \dot{\psi}\right) = (4, 0, 0, 0.3, 0, -0.5, 6, -3, 0, 0)\,. \tag{33}$$

The controller is now able to limit the precession still maintaining the rod upward as shown by Figure 6.

4 Conclusion

In this paper, we have proposed a state space model and a simple controller for the rodwheel. Since it is composed of one wheel, one motor and one rod, the rodwheel can be seen as the most elementary wheeled robot than can be built, *i.e*, it seems impossible to build a fast vehicle with less mechanical components. Due to its shape, with almost no front surface that could slow down the vehicle, we could expect to have a fast robot which makes long trips with few batteries. Moreover, the expected weight could be low enough to limit the danger is case of collision.

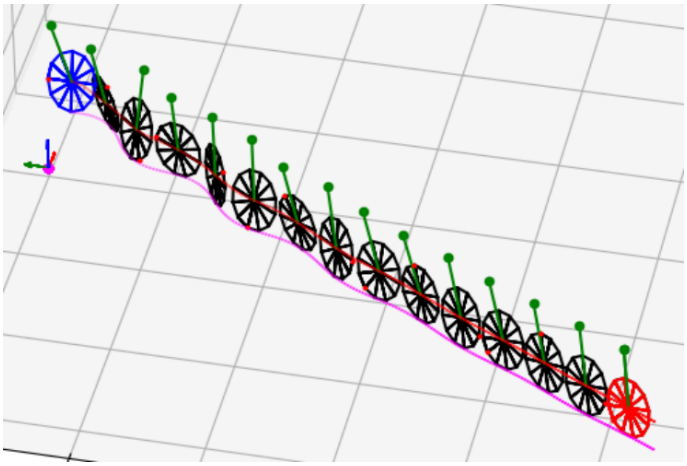

Fig. 6: The controller stabilizes the precession of the rodwheel

We have shown that it was possible to stabilize the robot and control the speed. Now, due to the singularity for $\theta = 0$ (which corresponds to the situation where the wheel goes straight ahead) the system becomes uncontrollable. We should thus add a other actuator such as a small inertial wheel fixed the end of the rod to be able to control the heading.

We have proposed a stability analysis which is based on simulation and symmetries, which is clearly not enough. The stability should be studied by a linearization to have a better understanding of the instabilities and to get a better tuning of the control parameters.

Since the rodwheel is considered here as a mobile robot, the heading control needs to be treated. An easy solution would be to add an additional actuated inertial wheel spinning perpendicularly with respect to the rod. From a control point of view, a more challenging question would be to control the heading of the robot without changing the mechanics, *i.e.* , changing the controller only.

The `Python` code associated to all examples can be found here:

https://www.ensta-bretagne.fr/jaulin/rodwheel.html

References

[1] P. Appell. Sur l'intégration des équations du mouvement d'un corps pesant de révolution roulant par une arête circulaire sur un plan horizontal; cas particulier du cerceau. *Rendiconti del Circolo Matematico di Palermo*, 1900. 1

[2] F. Boyer, M. Porez, and J. Mauny. Reduced dynamics of the non-holonomic whipple bicycle. *Journal of Nonlinear Science*, 28, 06 2018. 1

[3] P. Corke. *Robotics, Vision and Control*. Springer, Berlin Heidelberg, 2011. 1

[4] D. Esnault. Robot d'engagement gyrostabilisé de reconnaissance. *Rapport de PFE, ENSTA-Bretagne, Brest, France*, 2023. 1

[5] I. Fantoni and R. Lozano. *Non-linear control for underactuated mechanical systems*. Springer-Verlag, 2001. 1

[6] L. Jaulin. *Mobile Robotics*. ISTE editions, 2015. 2.2

[7] L. Jaulin. Modelisation of a rolling disk with sympy. *arXiv:2311.16624, math.NA*, 2023. 2.1, 2.5

[8] O. O'Reilly. The dynamics of rolling disks and sliding disks. *Nonlinear Dynamics*, 10(3):287–305, 1996. 1

[9] D.A. Wells. *Lagrangian Mechanics*. Schaum's outlines, London, 1967. 1