

# Magnetic mapping using a trailed magnetometer

Quentin Brateau<sup>1</sup>✉

<sup>1</sup> ENSTA Bretagne, Brest

The realization of a magnetic map of a terrain is a powerful tool especially used in mine warfare. This cartography can be done with a magnetometer, but the task is quite slow. This is why it is useful to use robots to make this cartography. The problem induced by this solution is the addition of magnetic disturbances related to the structure of the robot and its actuators. It is then possible to drag the magnetometer on a sled behind the robot.

Magnetic | Mapping | Control | Interval Analysis  
Correspondence: [quentin.brateau@ensta-bretagne.org](mailto:quentin.brateau@ensta-bretagne.org)

## Formalism

The system evolve on a field which should be mapped with the magnetometer. It follows the following state equations :

$$\begin{cases} \dot{x} = f(x(t), u(t)) & (\text{evolution equation}) \\ y^i = g(x(t_i)) & (\text{observation equation}) \end{cases} \quad (1) \quad (2)$$

where  $x \in \mathbb{R}^4$  is the state of the robot,  $u \in \mathbb{R}^2$  is the input of the system. As  $x$  and  $u$  continuously evolve with time, we define the trajectories of this system denoted by  $x(\cdot)$  and  $u(\cdot)$ . We could also define tubes enclosing these trajectories at every time denoted by  $\forall t, [x](t)$  and  $\forall t, [u](t)$ .

## Modelisation of the system

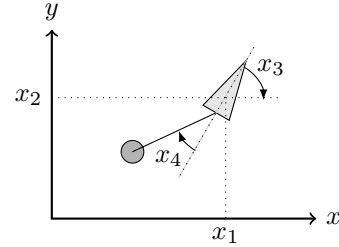
The system is composed of a vehicle that will drag a sledge that transports a magnetometer. The towing vehicle is a tank type vehicle, and the sled is attached to it with a rope. Assuming that the rope is constantly under tension, we are able to find the equations describing the dynamics of the system. By noting  $x$  the state of the system and  $u$  its inputs, the dynamics of the system is described by :

$$\dot{x} = f(x, u) = \begin{cases} \dot{x}_1 = \frac{u_1 + u_2}{2} \cdot \cos(x_3) & (3) \\ \dot{x}_2 = \frac{u_1 + u_2}{2} \cdot \sin(x_3) & (4) \\ \dot{x}_3 = u_2 - u_1 & (5) \\ \dot{x}_4 = -\frac{u_1 + u_2}{2} \cdot \sin(x_4) - \dot{x}_3 & (6) \end{cases}$$

Here  $x_1$ ,  $x_2$  and  $x_3$  are respectively the abscissa, the ordinate and the heading of the trailing robot,  $x_4$  is the angle of the sled to the tractor vehicle.

## Simulator

A python simulation was realized using VIBES in order to test the behavior of the system. A class *Tank* has been cre-



ated to instantiate a vehicle with its sled. Then the script integrates the evolution equation using Euler's method in order to obtain the state of the system  $x$  according to the  $u$  inputs. We could see that the behavior of the system seems correct and the model is faithful to reality. Moreover, the GNSS sensor and an accelerometer are simulated in order to enclose the real position in a box.

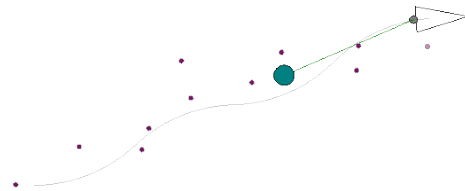


Fig. 1. Simulation of the system

## Reliable set of sled's angle

Considering that the rope remains continuously under tension while the robot is moving, then the evolution of the angle  $x_4$  of the sled relative to the towing vehicle is described by the *Differential Equation 6*.

We are able to find the trajectory of the sled by computing the interval containing the angle  $x_4$ , considering that initially  $x_4$  belongs to  $[-\pi/2; \pi/2]$ . Then by applying the *Differential Equation 6* on this interval, knowing the control vector  $u$ , we end up obtaining a fine interval framing the real angle  $x_4$ , independently of the initial angle as we can see on the **FIGURE 2**.

## Sled localization

By having a box enclosing the trailing robot, and an interval framing the angle  $x_4$ , we are able to obtain a box containing

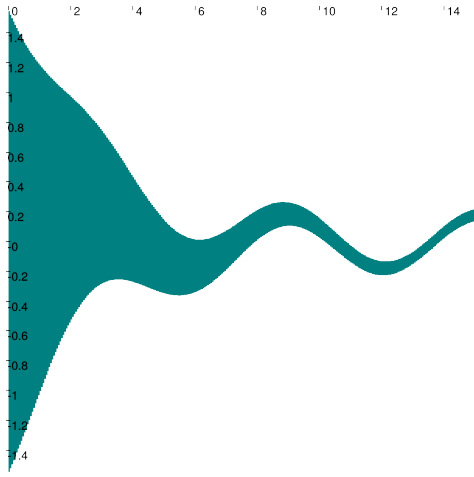


Fig. 2. Integration example using intervals

the sled, knowing the length of the rope.

This box has the shape of a pie sector. Equation 7 presents the position of the magnetometer as a function of the state of the system  $x_1$ ,  $x_2$  and  $x_4$ . Here all these variables are intervals. By applying a polar contractor to these intervals, we are able to obtain the intervals containing  $x_m$  and  $y_m$ .

$$\begin{cases} x_m = x_1 - L \cdot \cos(x_4) \\ y_m = x_2 - L \cdot \sin(x_4) \end{cases} \quad (7)$$

### Magnetometer range of measurement

By knowing a box containing the magnetometer and its measuring range, we are able to find all the points seen for sure and all the points that could be seen by the magnetometer. This allows us to recover the coverage of the area by the magnetometer.

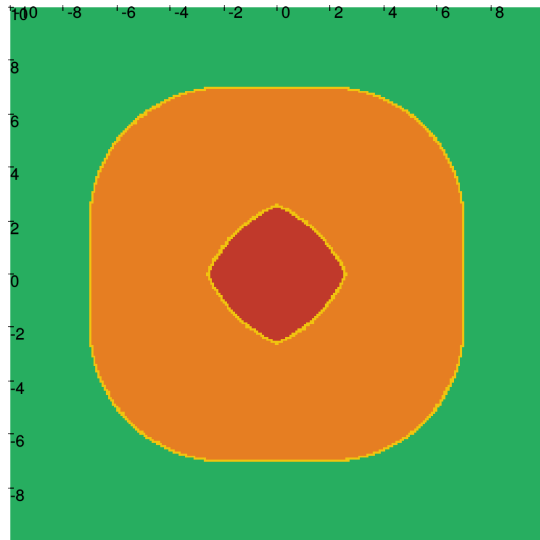


Fig. 3. Measurement range of the magnetometer using Thicksets

The FIGURE 3 shows us these two sets. The green set is the unseen area, the orange set is the area maybe seen and the red set is the area seen for sure by the magnetometer. The yellow set forms the uncertain boundary between the other

sets, which can be adjusted. Thus we have a way to know precisely the coverage of the mapping during the mission.

### Magnetometer Mapping

With a moving box enclosing the magnetometer at any time, this method allows us to recover the coverage of the area by the magnetometer. Thus we have a way to know precisely the coverage of the mapping during the mission, which depends on the uncertainty on the position of the magnetometer and on the maximum distance of each point of the field to be measured. An example of the covered area of a magnetometer on a field is presented by the FIGURE 4.



Fig. 4. Coverage of the map by the magnetometer using Thicksets

### Application example

In this part we will see on a practical case what we are able to do with this implementation. We place ourselves in the same conditions as before, i.e. we are in the case of a magnetometer dragged by a robot with tank type evolution equations and the rope must remain tight all along the mission.

Here we want our robot to follow the following Lissajous trajectory:

$$\forall t \in \mathbb{R}, \quad \mathcal{T}(t) = \begin{cases} x(t) = 20 \times \sin\left(\frac{2 \times t}{20}\right) \\ y(t) = 20 \times \sin\left(\frac{3 \times t}{20}\right) \end{cases} \quad (8)$$

$$\forall t \in \mathbb{R}, \quad \frac{d\mathcal{T}(t)}{dt} = \begin{cases} \frac{dx(t)}{dt} = 2 \times \cos\left(\frac{2 \times t}{20}\right) \\ \frac{dy(t)}{dt} = 3 \times \cos\left(\frac{3 \times t}{20}\right) \end{cases} \quad (10)$$

$$\forall t \in \mathbb{R}, \quad \frac{d^2 \mathcal{T}(t)}{dt^2} = \begin{cases} \frac{d^2 x(t)}{dt^2} = -\frac{4}{20} \times \sin\left(\frac{2 \times t}{20}\right) \\ \frac{d^2 y(t)}{dt^2} = -\frac{9}{20} \times \sin\left(\frac{3 \times t}{20}\right) \end{cases} \quad (12)$$

$$(13)$$

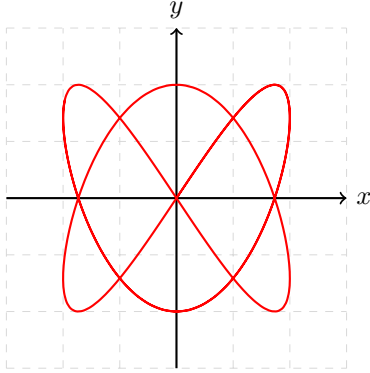


Fig. 5. Lissajous trajectory

The *control.py* file is able to generate the controls to drive the trailing robot. These commands are generated by feedback linearization by controlling the robot in speed and heading. These controls leads to the trajectory in the simulator as we could see on the FIGURE 6

Then the module *mapping.py* allows to process the tubes including the trajectory of the trailing robot and the sledge. The tubes are then displayed in order to check that they seem to stick to the trajectory of the system.

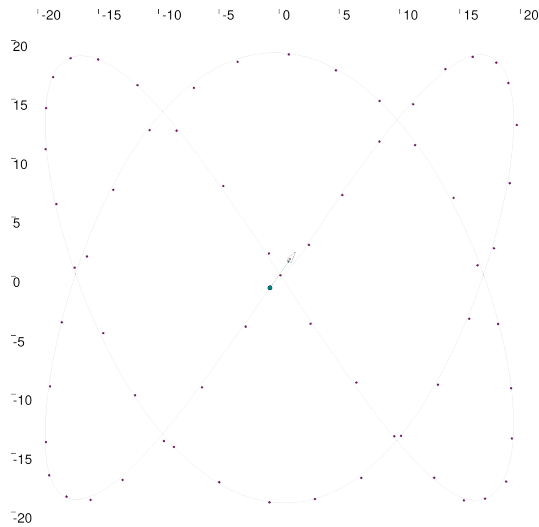


Fig. 6. Simulator's trajectory of the system

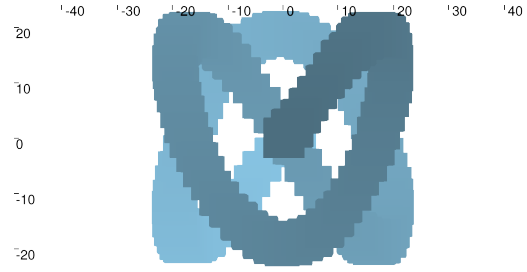


Fig. 7. Trailing vehicle enclosing tube

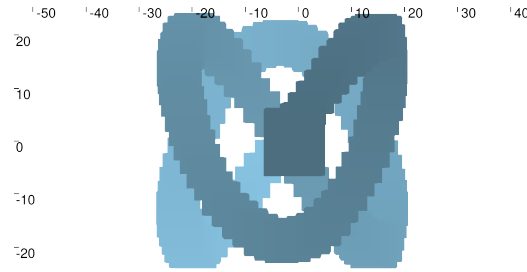


Fig. 8. Magnetometer enclosing tube

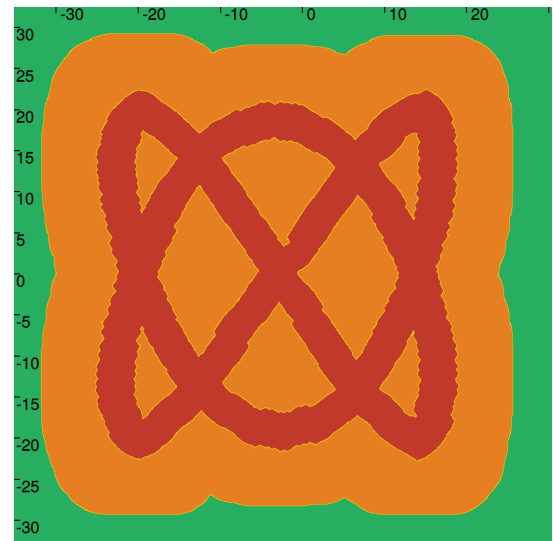


Fig. 9. Magnetometer enclosing tube