Sloshing supression

Governing equations and general knowledge base

Kai Bauer
kai.bauer2@kit.edu

Version 0.0.1
July 17, 2025

Contents

1 Introduction

1.1 About

This file is supposed to be a place to make more elaborate notes on the following topics:

- sloshing
- robotics
- control theory
- task planning
- trajectory planning

this file is under heavy construction

2 Sloshing

This section describes the phenomenon of liquid sloshing during container movement in more detail. It is also supposed to be able for usage as template and knowledge base for future publications.

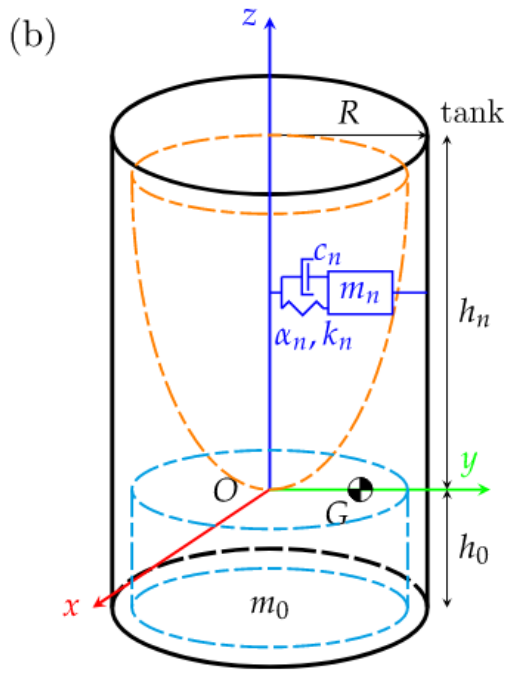

Figure 1: sketch stolen from Viet et al. [**viet_antisloshing_2024**]

2.1 General

The quoted literature for basics on the topic of sloshing modeling are Abramson [**abramson_dynamic_1966**]

2.2 Equations

General

Symbol	Description	Unit
R	radius of the tank	m
h	height of the liquid in the tank	m
g	gravitational acceleration	m/s^2
ξ_{1n}	root of derivative of the Bessel function first kind	-
m_F	total mass	kg
m_0	ridgid mass	kg
m_n	moving mass	kg

conservation of mass equates to:

$$m_F = m_0 + \sum_{n=1}^{\inf} m_n \tag{1}$$

the natural frequency of the sloshing motion is given by:

$$\omega_n^2 = \frac{g\xi_{1n}}{R} tanh\left(\xi_{1n}\frac{h}{R}\right) \tag{2}$$

giving the n-th sloshing mass as:

$$m_n = m_F \frac{2R}{\xi_{1n}h(\xi_{1n}^2 - 1)} tanh\left(\xi_{1n}\frac{h}{R}\right) \tag{3}$$

damping ratio:

$$\zeta_n = \frac{2.89}{\pi}\sqrt{\frac{\nu}{R^{3/2}g^{1/2}}} \tag{4}$$

The sloshing mass moves along a parabolic surface, with an attached nonlinear spring exerts forces $\alpha_n k_n x_n^{2w-1}$ and $\alpha_n k_n y_n^{2w-1}$ in the x and y direction respectively and $\alpha_n k_n r_n^{2w-1}$ in the radial direction for the radia generalised coordinate r_n.

Parabolic surface's analytical formula:

$$z_n = \frac{\omega_n^2}{2g}(x_n^2 + y_n^2) \tag{5}$$

Time derivative:

$$\dot{z}_n = \frac{\omega_n^2}{2g}(x_n\dot{x}_n + y_n\dot{y}_n) \tag{6}$$

EoM from Lagrange:

$$\begin{cases} \dfrac{d}{dt}\left(\dfrac{\partial \mathcal{T}}{\partial \dot{x}_n}\right) - \dfrac{\partial \mathcal{T}}{\partial x_n} + \dfrac{\partial \mathcal{V}}{\partial x_n} = -\dfrac{\partial \mathcal{D}}{\partial \dot{x}_n} \\ \dfrac{d}{dt}\left(\dfrac{\partial \mathcal{T}}{\partial \dot{y}_n}\right) - \dfrac{\partial \mathcal{T}}{\partial y_n} + \dfrac{\partial \mathcal{V}}{\partial y_n} = -\dfrac{\partial \mathcal{D}}{\partial \dot{y}_n} \end{cases} \tag{7}$$

$\mathcal{T}$ being the kinetic energy of the nth sloshing mass

$$\mathcal{T} = \frac{1}{2}m_n[(\dot{x}_0+\dot{x}_n)^2+(\dot{y}_0+\dot{y}_n)^2+\dot{z}_n^2] = \frac{1}{2}m_n\left[(\dot{x}_0+\dot{x}_n)^2 + (\dot{y}_0+\dot{y}_n)^2 + \frac{\omega_n^4}{g^2}(x_n\dot{x}_n + y_n\dot{y}_n)\right] \tag{8}$$

$\mathcal{V}$ being the potential energy (gravity and non linear spring)

$$\mathcal{V} = mgz_n + \int_0^{r_n} \alpha_n k_n r_n^{2w-1} dr_n = mg\frac{\omega_n^2}{2g}(x_n^2 + y_n^2) + \frac{\alpha_n k_n}{2w}(x_n^2 + y_n^2)^w \tag{9}$$

$\mathcal{D}$ is the energy dissipation based on Rayleigh

$$\mathcal{D} = \frac{1}{2}c_n(\dot{x}_n^2 + \dot{y}_n^2 + \dot{z}_n^2) = m_n\zeta_n\omega_n\left[\dot{x}_n^2 + \dot{y}_n^2 + \frac{\omega_n^2}{g}(x_n\dot{x}_n + y_n\dot{y}_n)\right] \tag{10}$$

Resulting EoM:

$$\begin{cases} (1+\frac{\omega_n^4}{g^2})\ddot{x}_n + \frac{\omega_n^4}{g^2}x_n y_n \ddot{y}_n + (2\omega_n\zeta_n(1+\frac{\omega_n^4}{g^2}x_n^2) + \frac{\omega_n^4}{g^2}x_n\dot{x}_n) \\ +(2\omega_n\zeta_n\frac{\omega_n^4}{g^2}x_n y_n + \frac{\omega_n^4}{g^2}x_n\dot{y}_n)\dot{y}_n + \omega_n^2 x_n(1+\frac{\alpha_n}{R^2}(x_n^2+y_n^2)) & = -\ddot{x}_0 \\ \\ \frac{\omega_n^4}{g^2}x_n y_n \ddot{x_n} + (1+\frac{\omega_n^4}{g^2}y_n^2)\ddot{y}_n + (2\omega_n\zeta_n\frac{\omega_n^4}{g^2}y_n + \frac{\omega_n^4}{g^2}y_n\dot{x}_n)\dot{x}_n \\ +(2\omega_n\zeta_n(1+\frac{\omega_n^4}{g^2}y_n^2) + \frac{\omega_n^4}{g^2}y_n\dot{y}_n)\dot{y}_n + \omega_n^2 y_n(1+\frac{\alpha_n}{R^2}(x_n^2+y_n^2)) & = -\ddot{y}_0 \end{cases} \tag{11}$$

the maximum sloshing height is given by:

$$\bar{\eta}_n = \frac{\xi_{1n}^2 h m_n}{m_F R}\sqrt{x_n^2 + y_n^2} = \frac{\xi_{1n}^2 h m_n}{m_F R} r_n \tag{12}$$

simplification: the sum of the sloshing heights caused by the higher modes is negligible compared to the first mode. Therefore the free surface shape $\eta(r,\theta)$ can be approximated by:

$$\bar{\eta}(r,\theta) = \sum_n \bar{\eta}_n \frac{J_1(\xi_{1n}\frac{r}{R})}{J_1(\xi_{1n})}\cos(\theta) \approx \bar{\eta}_1 \frac{J_1(\xi_{1n}\frac{r}{R})}{J_1(\xi_{1n})} cos(\theta) \tag{13}$$

dynamic equation of the sloshing model:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + D = Qu \tag{14}$$

with:

$$\mathbf{M}(\mathbf{q}) = \begin{bmatrix} (1+\frac{\omega_n^4}{g^2}x_n^2) & \frac{\omega_n^4}{g^2}x_n y_n & 0 & 0 \\ \frac{\omega_n^4}{g^2}x_n y_n & (1+\frac{\omega_n^4}{g^2}y_n^2) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; \mathbf{q} = \begin{bmatrix} x_n \\ y_n \\ x_0 \\ y_0 \end{bmatrix}; \mathbf{u} = \begin{bmatrix} u_x \\ u_y \end{bmatrix};$$

$$\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) = \begin{bmatrix} 2\omega_n\zeta_n(1+\frac{\omega_n^4}{g^2}x_n^2) + \frac{\omega_n^4}{g^2}x_n\dot{x}_n & 2\omega_n\zeta_n\frac{\omega_n^4}{g^2}x_n y_n + \frac{\omega_n^4}{g^2}x_n\dot{y}_n & 0 & 0 \\ 2\omega_n\zeta_n\frac{\omega_n^4}{g^2}x_n y_n + \frac{\omega_n^4}{g^2}y_n\dot{x}_n & 2\omega_n\zeta_n(1+\frac{\omega_n^4}{g^2}y_n^2) + \frac{\omega_n^4}{g^2}y_n\dot{y}_n & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix};$$

$$\mathbf{G}(\mathbf{q}) = \begin{bmatrix} \omega_n^2 x_n(1+\frac{\alpha_n}{R^2}(x_n^2+y_n^2)) \\ \omega_n^2 y_n(1+\frac{\alpha_n}{R^2}(x_n^2+y_n^2)) \\ 0 \\ 0 \end{bmatrix}; \mathbf{Q} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{15}$$

3 Robotics

3.1 Manipulator

Goal of the laboratory is to use one or more 6 or 7 DOF manipulators for basic and advanced laboratory tasks.

Modeling a robot involves studies about its kinematic behavior, namely the forces needed to produce a certain motion. This provides a description of how the links of the robot move in relation to each other. Two problems arise: the direct and inverse kinematics problem. The direct kinematics problem is to find the position and orientation of the end effector given the joint angles. The inverse kinematics problem is to find the joint angles given the position and orientation of the end effector. For serial Robots, like 6 DOF arms, the inverse kinematics problem is more complex than the direct kinematics problem.

3.1.1 Canidates

- KUKA LBR iiwa 7 R800
- UR7e - force-moment-sensor only in the gripper, joint torque mode ?
- Franka Reaserch
- Kinova Gen3
- Flexiv Rizon - direct force control

3.2 Platform

3.3 sensors

3.3.1 Tracking System

Camera solution for tracking the area of operation. Mostly done by triangulation of a point via multiple cameras. Common systems are provided by :

- Vicon
- OptiTrack
- Qualisys

3.3.2 Camera System

Robot endeffector mounted cameras for tracking the environment. Can be used for object detection and tracking, as well as for navigation. There different classes of cameras:

- RGB cameras
- Depth cameras
  - Basler ToF
- LIDAR

3.4 Enviroment

3.5 Simulation

Simulation of robotics behaviour is done primarily in the Gazebo simulator. The simulation is based on the Robot Operating System (ROS) and uses the ROS control framework to simulate the robot's behavior. As an alternative Simulation enviroment NVIDIA Isaac Sim is being considered.

3.5.1 Gazebo

Gazebo Sim is a widely used open source simulation enviroment. There is already lots of ROS2 support. This makes for a good starting point for the simulation of the robotic enviroment. Gazebo is available on local hardware and can be used without any special requirements.

Trajectory planning through the ROS2 MoveIt! package is possible.

3.5.2 NVIDIA Isaac Sim

Isaac Sim, due to its NVIDIA grafics driver requirement is (atm) only functional on the DPE workstation. The usage is therefor not as straight forward as Gazebo, which is available on local hardware. To access the NVIDIA Isaac Sim enviroment, one must first connect to the server via SSH ('username'@172.22.64.245). Adress the Gitlab repository at https://gitlab.kit.edu/kit/mvm/dpe/it/server-config for more detailed Information on how exactly to run Isaac sim.

Inside of the Isaac Sim Container, the enviroment needs to be set up. Namely the RWM and the path to the ROS2 bridge need to be set. Without the ROS2 bridge library path specification, the ROS2 bridge can not (easily?) be used in Isaac Sim.

```
export RMW_IMPLEMENTATION=rmw_cyclonedds_cpp
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/isaac-sim/exts/isaacsim.ros2.bridge/humble/lib
```

For ROS2 Communication the ROS2 Container has to be on the same network as the Isaac Sim Container. If this is not done due to the setup, the ROS2 Container can be started with the network attribute as follows:

```
docker run -it --network=host "ROS2_Container"
```

e.g. in current implementation (plus mounting the a local directory to the container):

```
docker run -it --network=host -v ~/data/autonomouslab/robotics_tests_git/ros2:/mnt/ros2
osrf/ros:humble-desktop
```

4 Tasks and inspiration

Sloshing free liquid transport. In extension supression of sloshing in multiple containers at once. Further tasks are to be determined, but are likely based on the usecase of the biological laboratory. Therefore sample preperation, transport and analysis are possible future tasks.

https://umi-gripper.github.io/

5 Control

First implimentions will make use of **Feed Forward Control**. with proper modeling of the Fluid behavior this control method should be sufficient to supress sloshing in the containers if the initial state of the fluid is known eg. at rest.

One goal is the usage of a **Closed Loop Control**. Possible methods might be basic **PID**, **model predictive control**. One of the main issues lies within the measurement of the output. For many tasks the output is neither well defined nor easily measurable. Therefore the control system has to be able to handle a large amount of uncertainty. Eg. in the case of sloshing control, the shape of the freed surface is very hard to measure. Finding alternative outputs of the system and using state estimation methods might be an option.

5.1 ROS2

ROS2 is the second generation of the Robot Operating System. It is a set of software libraries and tools that help you build robot applications. ROS2 is designed to be more flexible, scalable, and secure than its predecessor. It supports multiple programming languages, including C++ and Python, and provides a rich set of libraries for various tasks such as perception, control, and planning. Best suited for R&D and prototyping, but also used in industrie, military, and more.

most active version is ROS2 Humble Hawksbill, which is based on Ubuntu 22.04 LTS. Newest version at the writing of this document is ROS2 Jazzy, which is based on Ubuntu 24.04 LTS.

5.2 Industrial

EtherCAT (by ETG) is mostly used for robotics in industrial settings. Combines ease of use of Ethernet with the speed and low latency of fieldbus systems.