**SOS Optimization to Find Lyapunov Functions of Mass-Spring-Damper Systems**

Ben Walleshauser
11/17/2023

# 1 Introduction

Lyapunov functions are a powerful method to verify the stability of a system. Commonly they are used in control theory to demonstrate that a particular input provides asymptotic local or global stability. One particular challenge of using a Lyapunov method to prove stability of a controller is the formulation of the Lyapunov function for one's system, as there is usually no straightforward or algorithmic approach to doing so. This paper demonstrates the use of sum-of-squares (SOS) optimization to find the Lyapunov function for a given autonomous dynamical system. Sum-of-squares optimization allows for the explicit programming of inequality constraints to ensure the Lyapunov function is positive definite with respect to the system states and the derivative of the Lyapunov function is negative semi-definite. An equality constraint is also added to the problem as the equilibrium will be a fixed point. Two examples are given, namely the linear and nonlinear mass-spring-damper system. SOS optimization is performed with the use of the software PyDrake.

# 2 LaSalle's Invariance Principle

The autonomous system is defined by $\dot{\mathbf{x}} = f(\mathbf{x})$ where $\mathbf{x} \in \mathbb{R}^{n\times 1}$ corresponds to the system state. For a given Lyapunov function $V(\mathbf{x})$, LaSalle's invariance principle states that if the following conditions are met:

$$V(\mathbf{x}) \geq 0; \forall \mathbf{x} \neq \mathbf{0} \tag{1}$$

$$-\dot{V}(\mathbf{x}) = -\frac{\partial V}{\partial \mathbf{x}} f(\mathbf{x}) \geq 0; \forall \mathbf{x} \tag{2}$$

$$V(\mathbf{0}) = 0 \tag{3}$$

$$\lim_{\|\mathbf{x}\|\to\infty} V(\mathbf{x}) \to \infty \tag{4}$$

then the origin of the system is globally asymptotically stable. The challenging aspect of many controls problem is finding a suitable $V(\mathbf{x})$ for one's system, which is nontrivial for nonlinear systems. This is performed in this paper numerically using SOS optimization.

# 3 SOS Optimization

## 3.1 Problem Formulation

The candidate Lyapunov function chosen in this paper is a fixed-degree polynomial with unknown coefficients. For a system with $n$ states and fixed degree 2, this would be:

$$V_\alpha(\mathbf{x}) = \alpha_0 + \alpha_1 x_1 + ... + \alpha_n x_n + \alpha_{n+1} x_1^2 + ... + \alpha_{2n} x_1 x_n + ... + \alpha_{n!+n+1} x_n^2 \tag{5}$$

The goal of the optimization is to find the set of coefficients $\alpha$ that satisfy the conditions of LaSalle's invariance principle for a given system. A matrix $\mathbf{P}$ is said to be positive semi-definite if the following is true.

$$\mathbf{x}^\intercal \mathbf{P} \mathbf{x} \geq 0; \forall \mathbf{x} \tag{6}$$

But in general, a matrix $\mathbf{P}$ is said to be positive semi-definite if the following holds:

$$m(\mathbf{x})^\intercal \mathbf{P} m(\mathbf{x}) \geq 0; \forall \mathbf{x} \tag{7}$$

where $m(\mathbf{x})$ is a vector of monomials. This results is useful for finding the coefficients of a Lyapunov candidate, as if a polynomial $p(\mathbf{x})$:

$$p(\mathbf{x}) = c_1 + c_2 x_1 + c_3 x_1^2 + c_4 x_2 + c_5 x_2^2 + ... \tag{8}$$

can be written as:

$$p(\mathbf{x}) = m(\mathbf{x})^\intercal \mathbf{P} m(\mathbf{x}) \tag{9}$$

If $\mathbf{P}$ is positive semidefinite ($\mathbf{P}$ is often called the Gram matrix), then $p(\mathbf{x}) \geq 0; \forall \mathbf{x}$. Therefore, we can rewrite the search for the Lyapunov function in terms of the SOS constraint rather than the positive semi-definite condition, which is preferable as the SOS constraint is more computationally tractable [3]. The optimization problem is now as follows.

$$V_\alpha(\mathbf{x}) = \alpha_1 x_1 + ... + \alpha_n x_n + \alpha_{n+1} x_1^2 + ... + \alpha_{2n} x_1 x_n + ... + \alpha_{n!+n+1} x_n^2 \tag{10}$$

Where we want to find $\alpha$ s.t.:

$$V(\mathbf{x}) \text{ is SOS} \tag{11}$$

$$-\dot{V}(\mathbf{x}) \text{ is SOS} \tag{12}$$

Note that the radial constrain Eq. 4 is no longer included, as it is automatically satisfied by the inequality constraints and the form of the Lyapunov candidate. The equality constraint Eq. 3 is also automatically satisfied by making $\alpha_0 = 0$.

### 3.2 Solving SOS Programs

The goal is now to find the coefficients $\alpha$ comprising the two Gram matrices in the inequality constraints for $V(\mathbf{x})$ and $-\dot{V}(\mathbf{x})$ such the the Gram matrices are simultaneously positive semi-definite. Unfortunately, there is not a great deal of information as to how to explicitly solve these problems, but generally sum of squares programs reformulate the problem to be solved by semi-definite programs, which are then solved efficiently - commonly with the use of interior point methods[2]. Semi-definite program solvers include the likes of SeDuMi, SDPT3, CSDP, SPDNAL, SPDA, and MOSEK [2]. The optimization toolbox used in this paper is PyDrake, a Python and C++ toolbox supported by the Toyota Research Institute that can be used to model dynamical systems, run mathematical programs, and model multibody kinematics and dynamics [1]. For SOS optimization problems, PyDrake will preferably default to using either MOSEK or Clarabel depending on the form of the optimization problem. The Python code used in the subsequent sections follows the forms of the examples provided for the Underactuated Robotics course at MIT [4].

## 4 Linear Mass-Spring-Damper

The linear Mass-spring-damper has dynamics described by the following coupled differential equations.

$$f(\mathbf{x}) = \begin{bmatrix} \dot{x}_0 \\ \dot{x}_1 \end{bmatrix} = \begin{bmatrix} -2\zeta\omega x_0 - \omega^2 x_1 \\ x_0 \end{bmatrix} \tag{13}$$

Where $\zeta$ is the damping ratio and $\omega$ is the natural frequency. The goal is to find a Lyapunov function to prove that the system has global asymptotic convergence to the origin. The Lyapunov candidate now takes the following form with degree 2.

$$V_\alpha(\mathbf{x}) = \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_1 x_2 + \alpha_4 x_1^2 + \alpha_5 x_2^2 \tag{14}$$

The value of $\zeta$ is chosen to be 0.5 to correspond to an underdamped system, and the natural frequency $\omega$ is chosen to be 1 rad/s. By solving the optimization problem using PyDrake, the following Lyapunov function is found for the system:

$$V(\mathbf{x}) = 1.472520 x_0 x_1 + 1.951927 x_0^2 + 1.000000 x_1^2 \tag{15}$$

Note that this equation is equal to the sum of squares -

$$V(\mathbf{x}) = \begin{bmatrix} x_0 & x_1 \end{bmatrix} \begin{bmatrix} 1.951927 & 0.73626 \\ 0.73626 & 1.00000 \end{bmatrix} \begin{bmatrix} x_0 & x_1 \end{bmatrix} \tag{16}$$

where the Gram matrix has eigenvalues of $\lambda = 2.35267, 0.599254$, therefore confirming that $V(\mathbf{x})$ is positive semi-definite. The negative derivative of the Lyapunov function is therefore the following:

$$\dot{V}(\mathbf{x}) = -2.43133 x_0^2 - 3.37637 x_0 x_1 - 1.47252 x_1^2 \tag{17}$$

$$-\dot{V}(\mathbf{x}) = \begin{bmatrix} x_0 & x_1 \end{bmatrix} \begin{bmatrix} 2.43133 & 1.688185 \\ 1.688185 & 1.47252 \end{bmatrix} \begin{bmatrix} x_0 & x_1 \end{bmatrix} \tag{18}$$

The Gram matrix for $-\dot{V}(\mathbf{x})$ has eigenvalues of $\lambda = 3.70, 0.20$ therefore verifying that $-\dot{V}(\mathbf{x})$ is positive definite and confirming $V(\mathbf{x})$ is a valid Lyapunov function provided via the optimization. This ultimately tells us that the origin of the system $f(\mathbf{x})$ is globally asymptotically stable, which is intuitive as the damper dissipates energy until the system is at rest at the origin. The code can be ran below with the use of the DeepNote workspace attached below.

**Linear Case Deepnote Workspace (Link)**

## 5 Nonlinear Mass-Spring-Damper

The previous formulation for the linear mass-spring-damper demonstrated the use of SOS optimization to solve for the Lyapunov function for a linear system, which is typically easily found for linear systems by observing the total energy. The goal is to now find the Lyapunov function for the system with the nonlinear spring, described by the new $f(\mathbf{x})$ below.

$$f(\mathbf{x}) = \begin{bmatrix} \dot{x}_0 \\ \dot{x}_1 \end{bmatrix} = \begin{bmatrix} -2\zeta\omega x_0 - \omega^2 x_1^3 \\ x_0 \end{bmatrix} \tag{19}$$

The Lyapunov candidate is now chosen to have fixed degree of 4 to account for the nonlinear term in the dynamics. The values of $\zeta$ and $\omega$ are chosen to be $\zeta = 0.5$ and $\omega^2 = 1$ as in the linear case. By performing SOS optimization, the following Lyapunov function is found.

$$V(\mathbf{x}) = 0.722598 x_0 x_1 + 1.00000 x_0^2 + 0.361299 x_1^2 + 0.5000 x_1^4 \tag{20}$$

$$V(\mathbf{x}) = \begin{bmatrix} x_0 & x_1 & x_1^2 \end{bmatrix} \begin{bmatrix} 1.0000 & 0.361299 & 0 \\ 0.361299 & 0.361299 & 0 \\ 0 & 0 & 0.50000 \end{bmatrix} \begin{bmatrix} x_0 & x_1 & x_1^2 \end{bmatrix} \tag{21}$$

The Gram matrix for $V(\mathbf{x})$ has eigenvalues of $\lambda = 1.15, 0.50, 0.16$. The derivative of the optimized Lyapunov function is the following.

$$\dot{V}(\mathbf{x}) = -1.2774x_0^2 - 0.722598x_1^4 \tag{22}$$

$$-\dot{V}(\mathbf{x}) = \begin{bmatrix} x_0 & x_1^2 \end{bmatrix} \begin{bmatrix} 1.2774 & 0 \\ 0 & 0.722598 \end{bmatrix} \begin{bmatrix} x_0 & x_1^2 \end{bmatrix} \tag{23}$$

The Gram matrix for $-\dot{V}(\mathbf{x})$ of course has eigenvalues of $\lambda = 1.2274, 0.722598$. This analysis verifies the nontrivial nonlinear mass-spring-damper system is globally asymptotically stable at the origin for the given choice of $\zeta$ and $\omega$. The code can be ran below with the use of the DeepNote workspace attached below.

**Nonlinear Case DeepNote Workspace (Link)**

# 6 Conclusion

This work covers the application of SOS optimization to the solution of Lyapunov functions. It was found that the optimization technique returned valid Lyapunov functions for the linear and nonlinear mass-spring-damper systems. Future work may possibly involve applying this methodology to establish regions of attraction for various control laws, particularly the common linear-quadratic-regulator optimal control methodology. This strategy can also be used to possibly observe the effect of higher order terms in the quadratic-quadratic-regulator optimal control methodology.

# References


[1] Drake: Model-based design and verification for robotics. `https://drake.mit.edu/`. Accessed: 2023-11-26.

[2] Antonis Papachristodoulou, James Anderson, Giorgio Valmorbida, Stephen Prajna, Pete Seiler, Pablo Parrilo, Matthew M. Peet, and Declan Jagt. Sostools version 4.00 sum of squares optimization toolbox for matlab, 2021.

[3] Pablo A. Parrilo. Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization. 2000.

[4] Russ Tedrake. *Underactuated Robotics*. 2023.


# 7 Code

## 7.1 PyDrake Initialization in DeepNote

```
import matplotlib.pyplot as plt
import mpld3
import numpy as np
from IPython.display import Markdown, display
from pydrake.all import (
    Jacobian,
    MathematicalProgram,
    Solve,
    SymbolicVectorSystem,
    ToLatex,
    Variable,
    Variables,
```

```
)
from pydrake.symbolic import Polynomial

from underactuated import plot_2d_phase_portrait, running_as_notebook

if running_as_notebook:
    mpld3.enable_notebook()
```

### 7.2 Linear Case

```
#linear mass spring damper

def sos_optimize():
    prog = MathematicalProgram()
    x = prog.NewIndeterminates(2, "x")
    f = [-1*x[0] - x[1], x[0]]

    V = prog.NewSosPolynomial(Variables(x), 2)[0].ToExpression()
    print("Candidate:")
    display(Markdown("$V(x) = " + ToLatex(V) + "$"))
    prog.AddLinearConstraint(V.Substitute({x[0]: 0, x[1]: 0}) == 0)
    prog.AddLinearConstraint(V.Substitute({x[0]: 0, x[1]: 1}) == 1)
    Vdot = V.Jacobian(x).dot(f)

    prog.AddSosConstraint(-Vdot)

    result = Solve(prog)
    assert result.is_success()

    print("Solution:")
    display(
        Markdown(
            "$V(x) = "
            + ToLatex(
                Polynomial(result.GetSolution(V))
                .RemoveTermsWithSmallCoefficients(1e-5)
                .ToExpression(),
                6,
            )
            + "$"
        )
    )


sos_optimize()
```

### 7.3 Nonlinear Case

```
#nonlinear mass spring damper

def sos_optimize():
    prog = MathematicalProgram()
    x = prog.NewIndeterminates(2, "x")
    f = [-x[1]**3 - x[0], x[0]]

    V = prog.NewSosPolynomial(Variables(x), 4)[0].ToExpression()
    print("Candidate:")
    display(Markdown("$V(x) = " + ToLatex(V) + "$"))
    prog.AddLinearConstraint(V.Substitute({x[0]: 0, x[1]: 0}) == 0)
```

```
    prog.AddLinearConstraint(V.Substitute({x[0]: 1, x[1]: 0}) == 1)
    Vdot = V.Jacobian(x).dot(f)

    prog.AddSosConstraint(-Vdot)

    result = Solve(prog)
    assert result.is_success()

    print("Solution:")
    display(
        Markdown(
            "$V(x) = "
            + ToLatex(
                Polynomial(result.GetSolution(V))
                .RemoveTermsWithSmallCoefficients(1e-5)
                .ToExpression(),
                6,
            )
            + "$"
        )
    )


sos_optimize()
```