baja Documentation

Release -rc1

Huckleberry Febbo, Yingshi Zheng, Nicholas Renbergm, Brian Ar

February 10, 2017

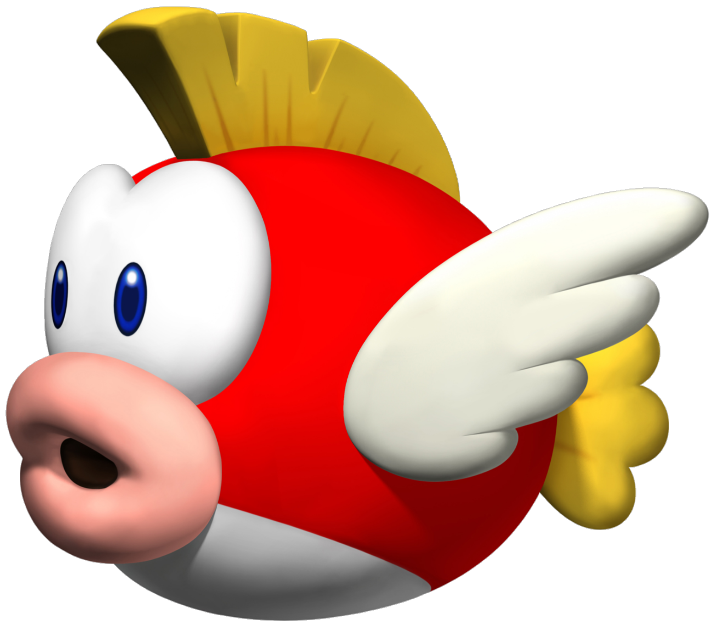
Contents

Contents:

CHAPTER 1

Weekly Meetings

1.1 Patrick



CHAPTER 2

ME 450 Project Requirements Spring 2017 - version #2

The overarching goal of this project is to apply basic engineering knowledge to augment an autonomous baja ground vehicle platform so that it can be controlled autonomously using both steering and speed commands. The project can be broken up into two main tasks:

1. Design and Manufacture
2. Experimental Testing

2.1 These tasks will be Completed in Parallel

The idea is that during *2) Experimental Testing* several of the students will become intimately familiar with current vehicle system by immediately beginning to *Test Steering System* as they set the stage for *Test Longitudinal Speed Systems* and simultaneously inform the *1) Design and Manufacture*. The idea is that the students should coalesce to complete the entire project as a whole and everyone should be familiar with the different facets of the project as they cultivate and hone their engineering skills.

Everyone should Be an Expert

Once the team is organized after the initial design phase, the team should work together to assign responsibility to individuals for specific parts of the project. For instance, if the team was designing an Automated Battery Exchange System For a Quadcopter, one guys would be the "gripper mechanism expert" another guy might be the "linear actuator expert" etc.

These two tasks are described in more detail below.

2.2 Meeting Schedule

As mentioned (via email on Jan 8th) the students are all responsible for these meeting. It is critical to complete these (weekly meeting notes) the team lead is responsible to manage the progress and tasks of the rest of the students and to create a Gannt chart to organize the overall project goals.

2.2.1 1) Design and Manufacture

Systems to control vehicle throttle and brake

In this task, students will be required to apply basic knowledge of electro-mechanical systems to remotely control the position of

- Select appropriate actuators (i.e. step motor, linear actuator, rotor motor). Considerations will include torque and speed requirements (which are also to be determined), safety, reliability and cost.
- A soft requirement is that the motors are back-drivable. This would be an extremely useful vehicle feature for the lab's research goals, but is not a hard requirement because it is a challenging task.
  - For those of you who do not know what back-drivability of a motor is check out this video.
- Determine actuator positions (after selecting one or more of the following; potentiometer, optical encoder, limit switch, etc.). Considerations for this task will include reliability, safety, and cost.
- Design and manufacture (or purchase) brackets and mounts to support all of the devices used in both throttle and brake systems. A basic stress analysis on these systems should be performed and the entire system should be moderately resilient to the elements.

2.2.2 2) Experimental Testing

Develop computer code to control the vehicle using steering angle and longitudinal speed

In this task, the students will be responsible for developing a rudimentary control algorithm (i.e. PID controller) to ensure that the vehicle can track predefined control signals.

- Design and manufacture (or purchase) any additional sensors and electrical equipment to accomplish this task.
- NOTE: to a large degree this task has already been accomplished (at least for the steering system). It is however up to the students to become familiar with it and assess the system's reliability and make any needed repairs or replacements.

Test Steering System

- Within the first month, students will become familiar with the current steering wheel actuator system and if needed replace any electronic or mechanical components.
- An adequate series of tests will be subsequently designed and carried out

Test Longitudinal Speed Systems

- As the **Design and Manufacturing** of these systems are carried out, the
- Both the throttle and brake systems will be tested to ensure that the they can operate the vehicle controls as expected. To do this basic knowledge of control and coding will be applied.
- An adequate series of tests will be subsequently designed and carried out

Remotely Control Vehicle

- For safety, the first series of test where the vehicle is being controlled remotely and with basic PID controller will be performed with the vehicle on an elevated platform (i.e. with the tires off of the ground).

- This is the next set of tests where the vehicle is controlled remotely as it navigates in a designated area.
- This is the next set of tests where the vehicle is controlled using the PID controller as it navigates in a designated area.

2.2.3 Current Platform:

The current vehicle has and actuator on the vehicle to control steering and there is station that is capable of teleoperating the vehicle using a steering wheel attached to an actuator for haptic feedback. However, there are significant dropouts in the control signals and the system is very rudimentary. Therefore, this project will be responsible for addressing these issues and creating a platform that has the mechanical and electrical components needed for autonomous and teleoperated driving.

CHAPTER 3

ME 450 Project Requirements Spring 2017 - version #1

The overarching goal of this project is to apply basic engineering knowledge to augment an autonomous baja ground vehicle platform so that it can be controlled autonomously using both steering and speed commands and add sensors to the vehicle that will allow the vehicle to perceive its environment and act accordingly. The project can be broken up into several tasks that are described below.

3.1 Design and Manufacture

3.1.1 Systems to control vehicle throttle and brake

In this task, students will be required to apply basic knowledge of electro-mechanical systems to remotely control the position of

- Select appropriate actuators (i.e. step motor, linear actuator, rotor motor). Considerations will include torque and speed requirements (which are also to be determined), safety, reliability and cost.
- Determine actuator positions (after selecting one or more of the following; potentiometer, optical encoder, limit switch, etc.). Considerations for this task will include reliability, safety, and cost.
- Design and manufacture (or purchase) brackets and mounts to support all of the devices used in both throttle and brake systems. A basic stress analysis on these systems should be performed and the entire system should be moderately resilient to the elements.

3.1.2 A mount to support a LiDAR device

In this task students will be required to design and manufacture a mounting system for a LiDAR device (to be selected by sponsors).

- Design and manufacture (or purchase) brackets and mounts to the LiDAR device.

Install GPS, IMU, Wheel Encoders, and Camera In this task students will be required to design and manufacture a mounting system for a GPS, IMU wheel encoders, and camera devices (to be selected).

- Design and manufacture (or purchase) brackets and mounts the sensors and devices.

3.1.3 Map steering wheel angle to angle of tire on ground

In this task students will be required to determine the relationship between the angle of the steering wheel and the angle of the tire at the ground.

3.1.4 Ensure that the following 10 states are accurately determined

Global vehicle position (X, Y, Z), vehicle velocity (lateral and longitudinal), yaw rate, yaw angle, steering angle (at tire ground interface), steering rate (at tire ground interface), and longitudinal jerk.

- Design and manufacture (or purchase) any additional sensors to accomplish this task.

3.1.5 Develop computer code to control the vehicle using steering angle and longitudinal speed

In this task, the students will be responsible for developing a rudimentary control algorithm (i.e. PID controller) to ensure that the vehicle can track predefined control signals.

- Design and manufacture (or purchase) any additional sensors and electrical equipment to accomplish this task.

3.1.6 Develop computer code to record all data from sensors

In this task, the students will be responsible for developing a data acquisition system to record all of the data and graph it (wher

  - Design and manufacture (or purchase) any additional sensors and electrical equipment to accomplish this task.

3.2 Experimental Testing

3.2.1 Testing Actuator Systems

- Both the throttle and brake systems will be tested to ensure that the they can operate the vehicle controls as expected. To do this basic knowledge of control and coding will be applied.
- Additionally, students will become familiar with the current steering wheel actuator system and if needed replace any electronic or mechanical components.

3.2.2 Testing Sensor and Remaining Electrical Systems

- The LiDAR device, GPS, IMU, wheel encoders, camera and any additional systems deemed necessary will be tested as well as data collecting and visualization software.
- Refine electrical system as needed, this may include anything from purchasing additional sensors to installing wire shielding.

3.2.3 Remotely and Autonomously Control Vehicle

- For safety, the first series of test where the vehicle is being controlled remotely and with basic PID controller will be performed with the vehicle on an elevated platform (i.e. with the tires off of the ground).
- This is the next set of tests where the vehicle is controlled remotely as it navigates in a designated area.
- This is the next set of tests where the vehicle is controlled using the PID controller as it navigates in a designated area.

3.3 Current Platform:

The current vehicle has and actuator on the vehicle to control steering and there is station that is capable of teleoperating the vehicle using a steering wheel attached to an actuator for haptic feedback. However, there are significant dropouts in the control signals and the system is very rudimentary. Therefore, this project will be responsible for addressing these issues and creating a platform that has the mechanical and electrical components needed for autonomous and teleoperated driving.

CHAPTER 4

Software and Resources

4.1 MATLAB

I used MATLAB to develop the work in my 2016 ASME paper. Great software, but julia is faster, free and just plain better:)

4.2 MPT3

- MPT3 is a MATLAB based optimization tool that comes out of The Automatic Controls Laboratory at ETH.
- Lots of other useful software that come out out ETH in Zurich, Switzerland and can be found aqui
- Additionally, I worked with Michal Kvasnica (a former memeber of The Automatic Controls Laboratory at ETH) developing Moving Obstacle avoidance code using MPT3.
    - The result of this work is a tool named OptiPlan and the code can be found here.

4.3 BARC

http://www.barc-project.com/

4.4 julia

4.4.1 Getting Started with julia

Extensive julia documentation is available here.

Building julia release 0.5 - current

Recently a new version of julia was released and it will be tested. The following directions also assume that julia was previously installed on the machine. If it was not, skip to step 3.

Remove current version of julia

1. get a graphical file manager with **ROOT ACCESS**:

```
gksu nautilus
```

Also, may need to:

```
sudo apt install gksu
```

- NOTE: be **very** careful in here!!

2. Type `ctrl+l` to type into location bar

- now navigate to where julia is and delete the binaries (or bin), ie. it might be in /usr/local/begin

Add new version current version of julia

3. unzip the .gz into the `\opt` folder (the place where by convention you'd put "optional" packages)

for instance:

```
sudo tar -xvf /home/febbo/Downloads/julia-0.5.0-linux-x86_64.tar.gz -C /opt
```

4. Check it:

```
febbo@febbo-HP-ZBook-17-G2:/opt/julia-3c9d75391c/bin$ ./julia
OR
febbo@febbo-HP-ZBook-17-G2:/opt/julia-3c9d75391c/bin$ /opt/julia-3c9d75391c/bin/julia
               _
   _       _ _(_)_     |  A fresh approach to technical computing
  (_)     | (_) (_)    |  Documentation: http://docs.julialang.org
   _ _   _| |_  __ _   |  Type "?help" for help.
  | | | | | | |/ _` |  |
  | | |_| | | | (_| |  |  Version 0.5.0 (2016-09-19 18:14 UTC)
 _/ |\__'_|_|_|\__'_|  |  Official http://julialang.org/ release
|__/                   |  x86_64-pc-linux-gnu

julia>
```

5. Now we need to create a symbolic link so that we can just type `julia` in the command line. There are three ways to do this (**just do the third option; it is permanent**):

   1. create an "alias"

   - create an alias using the following command..

     type:

```
         alias julia='/opt/julia-3c9d75391c/bin/julia'
```

   2. add `/opt/julia-3c9d75391c/bin` to your PATH

      - all directories contained in the PATH are accessible from **ANYWHERE**
      - you can see the current PATH variable by typing in your terminal..

     type:

```
         echo $PATH
```

      - you can redefine a variable in bash e.g. by doing..

     type:

```
         PATH=new definition here

           * (no spaces----> this is important)
```

- add your julia path to the existing path, you can do that by saying...

type:

```
PATH=/opt/julia-3c9d75391c/bin:$PATH

  * the $PATH contains the old value, so you're basically adding your folder and a colon and
```

The Problem with The Above Two Options is that:

With both adding to the path or creating an alias, is that these changes are **TEMPORARY** the minute you close your terminal and open it again, you'll see that these changes have disappeared try it and see!

- so to make it permanent, you actually have to do option 3

3. change a file called bashrc which is run every time a terminal starts

   - the file you need to edit should be /etc/bash.bashrc

     so type:

```
gksu gedit /etc/bash.bashrc
```

If it exists and it's the right file, you should see its contents inside gedit

- go to the very end of that file
  - add the alias line...

  type:

```
alias julia='/opt/julia-3c9d75391c/bin/julia'
```

and press enter!!

  - The enter is important; all commands should always end with a newline

6. Now close the terminal and type:

```
febbo@febbo-HP-ZBook-17-G2:~$ julia
               _
   _       _ _(_)_     |  A fresh approach to technical computing
  (_)     | (_) (_)    |  Documentation: http://docs.julialang.org
   _ _   _| |_  __ _   |  Type "?help" for help.
  | | | | | | |/ _` |  |
  | | |_| | | | (_| |  |  Version 0.5.0 (2016-09-19 18:14 UTC)
 _/ |\__'_|_|_|\__'_|  |  Official http://julialang.org/ release
|__/                   |  x86_64-pc-linux-gnu

julia>
```

So, it should be running!

7. If you are using Atom, make sure that you change the path in the config folder in settings:

```
juliaPath: "/opt/julia-3c9d75391c/bin/julia"
```

8. next time you download a new version of julia:

- simply extract it under /opt in the same way and either replace the old one

OR

- if you want to keep both, you can just update your alias in /etc/bash.bashrc to point to the new one

Building julia release 0.5 - old

1. Type:

```
sudo apt -rm julia
```

2. Also, julia can be completely removed by deleting the `~/.julia` folder.

Note: Can also remove PPA (according to this):

```
sudo apt install ppa-purge
```

- although this does not seem to be useful.

Fresh install instructions for UBUNTU:

1. Follow these instructions.

Or

B. Type this terminal:

```
sudo add-apt-repository ppa:staticfloat/juliareleases
sudo add-apt-repository ppa:staticfloat/julia-deps
sudo apt-get update
sudo apt-get install julia
```

Bleeding Edge Version - previously used

I found that julia is constantly being developed and that most of the developers do not make sure that every version is maintained, this can create issues when using particular packages. So, if you want to use the latest and greatest features, you might consider the bleeding edge version of julia.

Fresh install instructions for UBUNTU:

1. Follow these instructions.

Or

B. Type this terminal:

```
sudo apt-add-repository ppa:staticfloat/julianightlies
sudo apt-add-repository ppa:staticfloat/julia-deps
sudo apt-get update
sudo apt-get install julia
```

Warnings

If you are getting warnings like this:

```
WARNING: Deserialization checks failed while attempting to load cache from /home/febbo/.julia/lib/v0
WARNING: Module Lazy uuid did not match cache file.
INFO: Recompiling stale cache file /home/febbo/.julia/lib/v0.6/JuMP.ji for module JuMP.
WARNING: Deserialization checks failed while attempting to load cache from /home/febbo/.julia/lib/v0
WARNING: Module Lazy uuid did not match cache file.
INFO: Recompiling stale cache file /home/febbo/.julia/lib/v0.6/PyPlot.ji for module PyPlot.
```

```
WARNING: Deserialization checks failed while attempting to load cache from /home/febbo/.julia/lib/v0
WARNING: Module Conda uuid did not match cache file.
```

It is a precompilation failure; restart Julia

4.4.2 Useful Packages and Programs

This page includes details of the packages and programs that I am using in OCP.

Adding and Removing Packages in julia

A useful description of the syntax for adding and remove packages in julia can be found here.

All packages that I have on julia

I have many packages and to configure them all can be tricky, so I include a list of commands that you can copy and past into julia to get started. Make sure that you are restarting the both julia and the terminal after things are installed.

Basics

```
Pkg.add("DataFrames")
Pkg.add("IJulia")
Pkg.add("Parameters")
Pkg.add("PkgDev")
Pkg.add("AmplNLWriter")
#Pkg.clone("https://github.com/JunoLab/Juno.jl")
Pkg.add("HDF5")
Pkg.build("HDF5")
Pkg.add("SymPy")
Pkg.add("Jacobi")
```

Math

I use:

```
Pkg.add("DiffBase")
Pkg.clone("https://github.com/Keno/Polynomials.jl")
Pkg.add("DifferentialEquations")
Pkg.add("Dierckx")
Pkg.add("ImplicitEquations")
Pkg.add("Interpolations")
```

Optimization

I use:

```
Pkg.add("JuMP") # adds MathProgBase automatically
Pkg.add("Ipopt")
Pkg.add("CoinOptServices")
Pkg.add("NLopt")
Pkg.build("NLopt")
```

Plotting

Basically I use Plots.jl to interface with some of these:

```
Pkg.add("Conda")
ENV["PYTHON"]=""
Pkg.add("PyPlot")
Pkg.add("Plots")
Pkg.build("PyPlot")
Pkg.add("ImageMagick")
Pkg.add("GR")
Pkg.add("Plotly")
Pkg.add("StatPlots")
Pkg.add("PlotRecipes")
Pkg.add("UnicodePlots")
Pkg.add("Gadfly")
Pkg.add("RDatasets")
Pkg.add("Winston")
Pkg.add("PGFPlots")
Pkg.build("PGFPlots")
Blink.AtomShell.install()
import Conda # to fix pyplot!!!
Conda.add("qt=4.8.5")
```

My Packages

I started these:

```
Pkg.clone("https://github.com/huckl3b3rry87/VehicleModels.jl")
Pkg.clone("https://github.com/huckl3b3rry87/NLOptControl.jl")
```

Miscellaneous

probably do not need:

```
Pkg.clone("https://github.com/pwl/MovcolN.jl")
Pkg.add("Devectorize")
Pkg.add("FactCheck")
```

Useful Command After Installing Packages

Type:

```
Pkg.update()
```

Customizing Keybindings and Tab Completion

Info here

type:

```
import Base: LineEdit, REPL

const mykeys = Dict{Any,Any}(
  # Up Arrow
  "\e[A" => (s,o...)->(LineEdit.edit_move_up(s) || LineEdit.history_prev(s, LineEdit.mode(s).hist)),
  # Down Arrow
  "\e[B" => (s,o...)->(LineEdit.edit_move_up(s) || LineEdit.history_next(s, LineEdit.mode(s).hist))
)

function customize_keys(repl)
  repl.interface = REPL.setup_interface(repl; extra_repl_keymap = mykeys)
end
```

atreplinit(customize_keys)

4.4.3 Basic Pkg. Usage Notes

Optimization

Currently the only optimization tool that is being tested it IPOPT.

1. IPOPT

It is very easy to get going using IPOPT in julia using IPOPT.jl.

Derivatives

JuMP.jl is one of the most useful packages for solving the OCP because it takes **very** fast automatic derivatives and it allows the user to easily set up optimization problems. So, with this tool there is no need to write out the complicated Jacobian and Hessian functions.

The documentation for this package can be found JuMP docs.

Some useful Methods are found by clicking.

Some useful commands Query upper and lower bounds of all constraints in the model:

```
JuMP.constraintbound(m::Model)
```

MathProgBase.jl

MathProgBase.jl.

Polynomial Division

There was an issue when JuMP was sent a term with polynomial division. This section deals with the attempt to use the Polynomials.jl to take care of the polynomial division on the front end, before the expressions are sent to JuMP.

Basic Functionality:

```
using Ploynomials
Poly([1,0,3,4])
Poly(1 + 3x^2 + 4x^3)
```

Division Functionality:

```
P1 = Poly([1,2,3,5,7])
P2 = Poly([1,0,3])
P3 = div(P1,P2)
Poly(0.2222222222222218 + 1.6666666666666667x + 2.3333333333333335x^2)
```

Plots.jl

A very powerful plotting tool is Plots.jl. It took some time to get everything working because I was not using the same versions of the packages as the developers, but in the end it was work the time. It you run the code that I have listed below you should not have to deal with the setup issues that I had.

With PGFPlots:

```
sudo apt-get install pdf2svg
```

http://nbviewer.jupyter.org/github/sisl/PGFPlots.jl/blob/master/doc/PGFPlots.ipynb

Problem 1

I wasted a bunch if my time, could not recreate in a simpler example, but basically, when `plot()` should have worked it failed and when I changed the order of the terms in `plot()` it works.

Solution 1

EX:

```
for i in 1:k
#            plot!(dfs[i][:t],dfs[i][:SA]*180/pi,w=w2,label=label_string[i])
             plot!(dfs[i][:t],dfs[i][:SA]*180/pi,label=label_string[i],w=w2)
end
```

Problem 2

Segfault when attempting to plot after Conda update to Segfault with qt >=4.8.6 on ubuntu

I was running into an issue with Plots.jl after doing a `Pkg.update()`:

```
   _       _ _(_)_     |  A fresh approach to technical computing
  (_)     | (_) (_)    |  Documentation: http://docs.julialang.org
   _ _   _| |_  __ _   |  Type "?help" for help.
  | | | | | | |/ _` |  |
  | | |_| | | | (_| |  |  Version 0.6.0-dev.508 (2016-09-06 20:36 UTC)
 _/ |\__'_|_|_|\__'_|  |  Commit b1f1525 (26 days old master)
|__/                   |  x86_64-linux-gnu

julia> using Plots
julia> plot(rand(4,100))
signal (11): Segmentation fault
while loading no file, in expression starting on line 0
unknown function (ip: 0x32735)
Allocations: 14606607 (Pool: 14605017; Big: 1590); GC: 25
Segmentation fault (core dumped)
```

Solution 2

Change back to old Conda:

```
import Conda
Conda.add("qt=4.8.5")
using PyPlot
plot(rand(10))
```

Check out this link.

DifferentialEquations.jl

Useful Commands

all of the timepoints:

```
sol[:]
```

Many times you might want to use a good interpolation. For example, the plots use something like:

```
[sol(t) for t in linspace(t0,tend,100)]
```

To get 100 points from time t0 to tend for the first component:

```
[sol(t)[1] for t in linspace(t0,tend,100)]
```

The array of timepoints for component j:

```
[sol[i][j] for i in 1:length(sol)]
```

Parameters.jl

Great package for working with parameters. More info can be found at this link.

4.4.4 Making Your Own Packages

To create packages and modules click this to get started.

Or type:

```
using PkgDev
PkgDev.generate("VehicleModles","MIT")
```

Possible Next Steps:

- Make a github repository with this name (plus a `.jl` at the end of the name) github.com
  - Don't add a `README.MD` automatically using github, there will be a conflict if you will make one from the using `sphinx-quickstart`
    - Or just pick one, don't do both!
- Make some documentation *Sphinx*
  - Or, if everything is setup, type:

```
sphinx-quickstart
```

  - Then:

```
git remote add origin git@github.com:huckl3b3rry87/new_repo.jl
```

To tag:

```
using PkgDev
PkgDev.tag("VehicleModles")
```

To view the package:

```
$ cd ~/.julia/v0.6/VehicleModles && git show --stat
```

To recompile a package:

```
Pkg.test()
```

To go into the package directory:

```
cd(Pkg.dir("LiDAR"))
```

In Juno, the workflow for building a package, you can do a `Ctrl+J` `Ctrl+k` to quit the current process and then use `using PKG` again.

it'll be in your lib folder and all setup to be used with using:

```
Pkg.test
Pkg.dir
etc
```

Then in order to run `Pkg.update()` without getting this error:

```
ERROR: Update finished with errors.
=> Package NLOptControl cannot be updated.
GitError(Code:ERROR, Class:Merge, There is no tracking information for the current branch.)
etc....
```

Which is talked about here, you have to:

```
julia> Pkg.checkout("NLOptControl")
INFO: Checking out NLOptControl master...
INFO: Pulling NLOptControl latest master...
INFO: No packages to install, update or remove
```

Making Modules

look here

Directories

Try:

```
@__FILE__
```

link

4.4.5 Macros

Detailed information on macros in julia is found here.

@def

Given some parameters:

```
using Parameters

@with_kw immutable Vpara @deftype Float64
    m  = 2.6887e+03
end
pa = Vpara(); # initialize parameter set
```

Instead of unpackaging the same parameters each time in a nested function like this:

```
function outer_f(pa::Vpara)
  num = zeros(Float64, (10,1))
  for i in 1:10
   num[i] = inner_f(pa::Vpara,i)
  end
  return num
end
```

with:

```
function inner_f(pa::Vpara,i)
  @unpack_Vpara pa
   m + i + 0.1
end
```

We define a macro as:

```
macro def(name, definition)
  return quote
    macro $name()
      esc($(Expr(:quote, definition)))
    end
  end
end
```

then redefine `inner_f1()` as:

```
@def inner_f2 begin
  m + i + 0.1
end
```

We also need to modify the `outer_f()` as:

```
function outer_f(pa::Vpara)
  @unpack_Vpara pa
  num = zeros(Float64, (10,1))
  for i in 1:10
    num[i] = @inner_f2
  end
  return num
end
```

The `@def` macro is functionally equivalent to copying and pasting the contents of inner_f() into outer_f().

4.4.6 JuliaBox and Jupyter Notebooks

A way to run julia online without installing anything is to use JuliaBox with Jupyter notebooks.

The examples will be demonstrated using this tool.

Jupyter Notebooks

If you have IJulia installed you can run the examples using the following commands in julia:

```
using IJulia

# a few examples of changing the path
notebook(dir = Pkg.dir("VehicleModels")*"/examples")
notebook(dir="/home/febbo/Documents/workspace/OCP/examples")
```

More information can be found ' here <https://github.com/JuliaLang/IJulia.jl>'_.

4.4.7 julia notes

Using kwargs...

function:

```
function test(A; kwargs...)

    kw = Dict(kwargs)

    # if there was nothing passed -> set default
    if !haskey(kw,:mode); kw = Dict(:mode => :default) end
    mode = get(kw, :mode, 0);

    if mode == :default
      B = 10
    elseif mode == :LGRM
      B = A
    else
      print("pick a mode","\n")
    end
    return B
end

B=test(2)
B=test(2;(:mode=>:LGRM))
```

4.5 git

4.5.1 Website links

- https://github.com/
- The Pro Git book is available here!

4.5.2 Useful Commands

add everything to the commit (including new file and files that were deleted):

```
git add -A
```

commit all of the changes:

```
git commit -m "some message about what you did"
```

push to remote account:

```
git push origin master
```

view current tags:

```
git tag
```

making a new tag:

```
git tag -a V0.0.1 -m " new version 0.0.1"
```

committing a tag:

```
git push origin master --tags
```

checkout a tag:

```
git checkout -b [branchname] [tagname]
```

see which branch you are on:

```
git branch
```

connecting to github:

```
git remote add origin git@github.com:username/new_repo
```

- making a branch look here

Then make a new repository using the interweb

- Check out this link for more info.

Caching your github password:

```
git config --global credential.helper 'cache --timeout=3600'
# Set the cache to timeout after 1 hour (setting is in seconds)
```

4.5.3 Another way to connect to github it using ssh

Initially the error was:

```
febbo@febbo-HP-ZBook-17-G2:~/.julia/v0.5/VehicleModels$ git push origin master
Permission denied (publickey).
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
```

- This was obtained when initially setting up the git repositories in julia after cloning a package and trying to push modifications back up to the remote repository.
- Information on this can be founds at , or by following the two steps a fix may be obtained:
  1. Make an ssh key and add it to github, following.
  2. Check out this, or use the following commands:

  - A program to hold private keys for public authentication.

    type:

```
ssh-agent
```

  - Initially the agent does not hold any private keys.

    So run:

```
ssh-add
```

4.5.4 Mistakes I Made

- Make sure that you are working on the master branch!
  - Do not check out a tag and start making changes only to realize that you are not on the master branch!
- Trying to connect to github using ssh
  1. Create a github repository, with the name (for example: huckl3b3rry87/LiDAR.jl)
  2. Then

  Type this in the terminal:

```
febbo@febbo-HP-ZBook-17-G2:~/.julia/v0.5/LiDAR$ git remote add origin git@github.com:huckl3b3rry
```

  3. Then

  Try this:

```
febbo@febbo-HP-ZBook-17-G2:~/.julia/v0.5/LiDAR$ git pull master
```

  4. Next

  Get this:

```
fatal: 'master' does not appear to be a git repository
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
```

  Next we are going to test the ssh connection

  5) Attempt to ssh to GitHub By typing:

```
febbo@febbo-HP-ZBook-17-G2:~/.julia/v0.5/LiDAR$ ssh -T git@github.com
Hi huckl3b3rry87! You've successfully authenticated, but GitHub does not provide shell access.
```

  6) realize that you messed up by typing:

```
git pull master
```

  and not:

```
git pull origin master
```

4.6 Linux

4.6.1 Useful Linux Programs with julia

In the Linux terminal:

```
sudo dpkg --install atom-amd64.deb
sudo apt install ffmpeg
sudo apt-get install cmake bar time binutils make libssl-dev gfortran libunwind8-dev gcc g++ #clang c
sudo apt-get install gsfonts-x11
sudo apt-get install notepadqq
sudo apt-get update
sudo apt-get install gfortran
sudo apt-get install libnlopt0
sudo apt-get install openssh-server
sudo apt install ipmiutil
sudo apt-get install hdf5-tools
apm install latex # Atom package manager for latex!
sudo apt-get install pdf2svg # for PGFPlots
pip install sphinxcontrib-bibtex
```

4.6.2 Java

First I tried a few things including:

```
sudo apt-get install default-jre
sudo apt-get update
```

And:

```
sudo add-apt-repository ppa:openjdk-r/ppa
sudo apt-get update
sudo apt-get install openjdk-7-jre
```

This messed stuff up and then I deleted all of my old java versions using:

```
sudo natilus
```

Ended up needing to follow *this advice <http://askubuntu.com/questions/251213/unable-to-install-default-jdk* >:

```
sudo add-apt-repository ppa:webupd8team/java
sudo apt-get update
sudo apt-get install oracle-java8-installer
java -version
sudo apt-get install oracle-java8-set-default
```

This finally worked and I was able to run JabRef with no problem using:

```
java -jar JabRef-3.8.jar
```

If something is broken:

```
sudo apt-get -f install
```

4.6.3 Basic Linux Commands

Never run an executable in root!

Some more info can be found on ‘ this page <https://www.cyberciti.biz/faq/tar-extract-linux/>‘.

To unpack a tar file:

```
tar -xvf file.tar
```

To extract a .tar.gz:

```
tar -xzvf file.tar.gz
```

Unzip:

```
unzip stuff.zip -d /destination/folder
```

directory comment:

```
$HOME = /home/febbo
```

Run a jar file: java -jar lad

4.6.4 Issues and Fixes

1. Trying to uninstall sphinx so that I could get a newer version without bugs:

```
febbo@febbo-HP-ZBook-17-G2:~/Desktop/useful downloads$ pip uninstall sphinx
Cannot remove entries from nonexistent file /home/febbo/anaconda3/lib/python3.5/site-packages/easy-in
```

The problem: This probably won’t work:

```
conda update setuptools
```

The fix, was to download `ez_setup.py` from this link and run it in the terminal:

```
febbo@febbo-HP-ZBook-17-G2:~$ python ez_setup.py
```

Then I could uninstall Sphinx-1.4.1:

```
febbo@febbo-HP-ZBook-17-G2:~$ pip uninstall sphinx
Uninstalling Sphinx-1.4.1:
/home/febbo/anaconda3/bin/sphinx-apidoc
/home/febbo/anaconda3/bin/sphinx-autogen
/home/febbo/anaconda3/bin/sphinx-build
/home/febbo/anaconda3/bin/sphinx-quickstart
/home/febbo/anaconda3/lib/python3.5/site-packages/Sphinx-1.4.1-py3.5.egg
Proceed (y/n)? y
Successfully uninstalled Sphinx-1.4.1
```

There was reason I needed to uninstall this version of Sphinx was because the citations where not working.

Finally I installed Sphinx-1.5.2:

```
#Install from newest dev version in stable branch::
pip install git+https://github.com/sphinx-doc/sphinx@stable
```

4.7 Different Text Editors

I tried a few different text editors, but this one I really like and it is very similar to coding in MATLAB. It is a tool called JUNO that basically links a nice text editor called Atom to julia, so that you can run julia interactively.

4.7.1 Atom

Atom Flight Manual

Useful things:

spell check <https://github.com/atom/spell-check>:

```
ctrl+shift+;
```

Markdown Preview Package:

```
ctr+shift+m
```

https://github.com/atom/markdown-preview

Subscripts:

```
variable\_1+tab
```

Using $/LaTeX$ with Atom. It was suggested to use *Tex Live <https://www.tug.org/texlive/acquire-netinstall.html>* Where the *quick install instructions are here<https://www.tug.org/texlive/quickinstall.html>* https://atom.io/packages/latex

4.7.2 TeX Live

I use this with Atom.

https://www.tug.org/texlive/ https://atom.io/packages/latex

After installing TeX Live: * Add /usr/local/texlive/2016/texmf-dist/doc/info to INFOPATH. * Add /usr/local/texlive/2016/texmf-dist/doc/man to MANPATH (if not dynamically found). * Most importantly, add /usr/local/texlive/2016/bin/x86_64-linux to your PATH for current and future sessions.

To do this change the bashrc file:

```
gksu gedit /etc/bash.bashrc
```

Paste this at the end:

```
PATH=/usr/local/texlive/2016/bin/x86_64-linux:$PATH; export PATH # make sure that there are NO spaces
MANPATH=/usr/local/texlive/2016/texmf-dist/doc/man:$MANPATH; export MANPATH
INFOPATH=/usr/local/texlive/2016/texmf-dist/doc/info:$INFOPATH; export INFOPATH
```

Then in Atom, go to settings and add the path to the texlive:

```
/usr/local/texlive/2016/bin/x86_64-linux
```

4.7.3 Potential Issues

- If the julia binary is not where it belongs (for instance I put it in my /opt/... folder)
  - Change the julia `.config` file (in Atom) to the proper folder (where julia binaries are)
- **Otherwise**
    - leave it as : juliaPath: "julia"

I had to:

```
Pkg.clone("https://github.com/JunoLab/Juno.jl")
Pkg.update()
```

Make sure you close both atom and julia after installation. Now you should be able to run julia from Atom!

4.8 Creating Documentation

4.8.1 Sphinx

1. Installation

Or in the terminal run:

```
pip install sphinx
```

1.b. Uninstall

2. Getting Started

   - an awesome video introduction
   - toctree
   - basic commands and syntax
   - useful resource
   -
   - Common commands:
     - To start documenting:

```
        sphinx-quickstart
```

     - To manually build documentation:

```
        make html
```

     - To clean out an old build folder when things have changed significantly:

```
        make clean
```

     - **changing the themes** In `conf.py` change:

```
            html_theme = 'haiku'
```

       Or a my main one:

```
            import sphinx_rtd_theme
            html_theme = "sphinx_rtd_theme"
            html_theme_path = [sphinx_rtd_theme.get_html_theme_path()]
```

       * here is an awesome theme

3. Some issues that I ran into:

- Make sure that the "toctree" is indented by 3 space characters. I listed the .rst files by 4 space characters and this created an issue. To resolve the issue, you need to have the same indentation level.

- When cross referencing things in the document make sure you skip a space when you define the ref.
  - like this

  works:

```
.. _ploy_div:

Polynomial Division
-------------------
```

  - not like this

  fails:

```
.. _ploy_div:
Polynomial Division
-------------------
```

4. Setting up a server to build the documentation when a change is detected.

Installation:

```
pip install sphinx-autobuild
```

Then go into your main directory and type:

```
sphinx-autobuild docs docs/_build/html
```

Or the directory that contains `conf.py` and type:

```
sphinx-autobuild . _build_html
```

Then visit the website: http://127.0.0.1:8000/

- This will show you the live changes (after each save!!)

More information can be found using this resource.

4.8.2 Read The Docs

Read the Docs is useful resource to host the webpage.

Some Issues:

- The git repository for OCP is private and this website only hosts public repositories.
- Had to remove the last line here in `conf.py` for the code to work with ReadtheDocs.

like this:

```
extensions = [
    'sphinx.ext.autodoc',
    'sphinx.ext.doctest',
    'sphinx.ext.intersphinx',
    'sphinx.ext.todo',
    'sphinx.ext.coverage',
    'sphinx.ext.mathjax',
    'sphinx.ext.ifconfig',
    'sphinx.ext.viewcode',
    'sphinxcontrib.bibtex',
]
#     'sphinx.ext.githubpages',
```

- Had to make sure that the name of the project was correct
- Had to make sure that the webhook was activated on github

4.8.3 MkDocs

Another nice looking tool is MkDocs.

4.8.4 Documentor.jl

Another nice looking tool is here.

4.9 LaTeX

LaTeX is an incredible document typesetting tool, use it.

Beamer - for making slides!

4.10 Manging References

Properly managing references is a critical habit and these are some of the useful software tool that I use to manage my references.

4.10.1 Using Sphinx BibTex extension

Sphinx BibTex extension

- To install

  Do this:

```
 pip install sphinxcontrib-bibtex

 In ``conf.py`` add:
 ::

   extensions = ['sphinxcontrib.bibtex']

* MAKE SURE * that it is *added* to the rest of extensions!! Not just at the top, it will be rem

Like this:

::

  extensions = [
      'sphinx.ext.autodoc',
      'sphinx.ext.doctest',
      'sphinx.ext.intersphinx',
      'sphinx.ext.todo',
      'sphinx.ext.coverage',
      'sphinx.ext.mathjax',
      'sphinx.ext.ifconfig',
      'sphinxcontrib.bibtex',
      ]
```

Also, you can avoid these errors on readthedocs.com:

```
python /home/docs/checkouts/readthedocs.org/user_builds/nloptcontroljl/envs/latest/bin/pip install --
Could not open requirements file: [Errno 2] No such file or directory: 'pip install sphinxcontrib-bib
You are using pip version 8.1.2, however version 9.0.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
Command time: 0s Return: 1
```

By typing:

```
requirements.txt
```

Into the Advanced Settings Page and making a requirements.txt file with:

```
pip install --upgrade pip
pip install sphinxcontrib-bibtex
```

More on this is *here <https://github.com/huckl3b3rry87/NLOptControl.jl/issues/2#issuecomment-268052941>*

Example: To cite:

```
according to :cite:`someone` yada yada..
```

Then, at the end of the document include:

```
.. bibliography:: references.bib
```

4.10.2 JabRef

BibTex files can be managed with a free software called JabRef

4.10.3 EndNote

I still use this tool, but it costs money and I do not have it on Ubuntu.

4.11 Transferring Files To A Website

http://www.umich.edu/~umweb/how-to/homepage.html

https://docs.oracle.com/cd/E26502_01/html/E29001/remotehowtoaccess-14.html

type:

```
sftp sftp.itd.umich.edu
```

Or just do:

```
scp -r /home/febbo/Documents/workspace/OCP/docs/_build/html sftp.itd.umich.edu:/afs/umich.edu/user/f/
```

Setting Permissions:

https://mfile.umich.edu/?path=/afs/umich.edu/user/f/e/febbo/Private/html

4.12 Community Help

There is a tremendous community of people online that will help you with your projects. I have helped when I can, but largely other more experienced people usually help me. Below I have listed a few of the communities that I have been involved with.

4.12.1 Stackoverflow

A more general venue for asking questions to the online community of experts in various feilds.

4.12.2 GitHub

If you have issues with peoples software, you can submit an issue.

4.12.3 Gitter

Basicaly an instant message service for developers. I have used it for julia, plots.jl, and juno.jl.

4.13 Making A Thesis

https://github.com/jterrace/sphinxtr

CHAPTER 5

Code Development

5.1 Packages

5.1.1 VehicleModels.jl

This is a package that I created and it can be downloaded from here.

- The purpose of this package is to allow the easy collaboration and development of vehicle models.

The examples for this package are available by typing:

```
using IJulia
notebook(dir = Pkg.dir("VehicleModels")*"/examples")
```

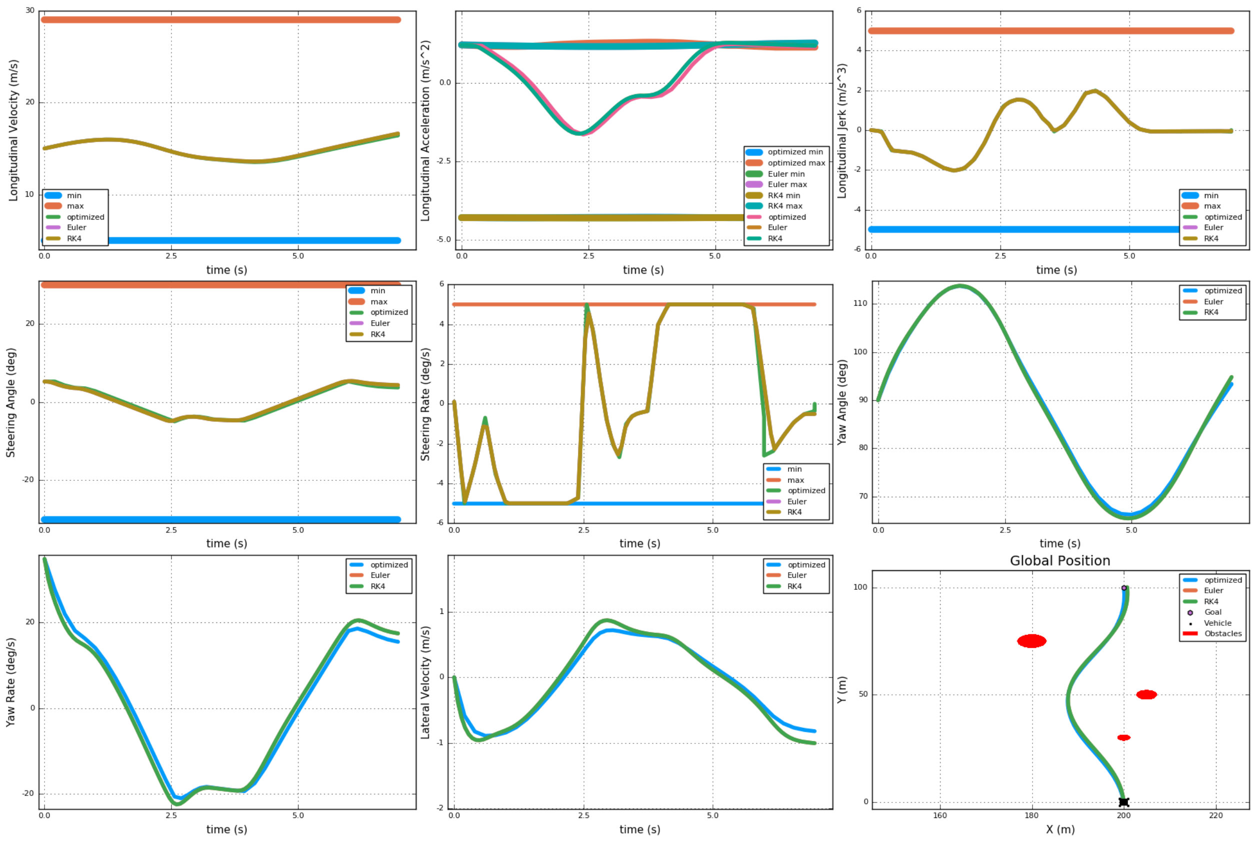
Three Degree of Freedom Model

In this example only the control signals (SR and Jx) are used to solve the differential equations using Euler and RK4 techniques and the results are compared to the optimized signals.

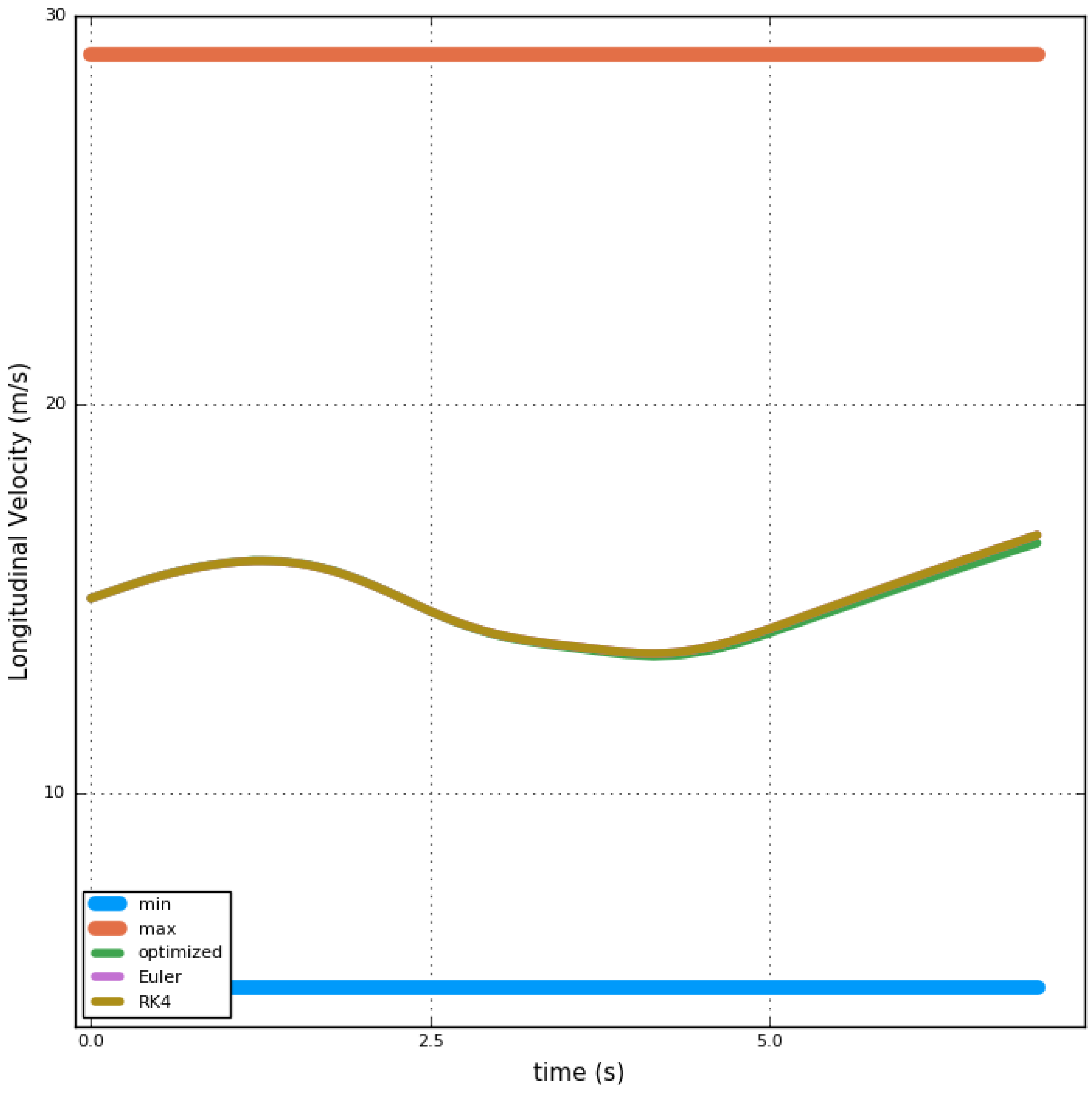
Notes & Conclusions:

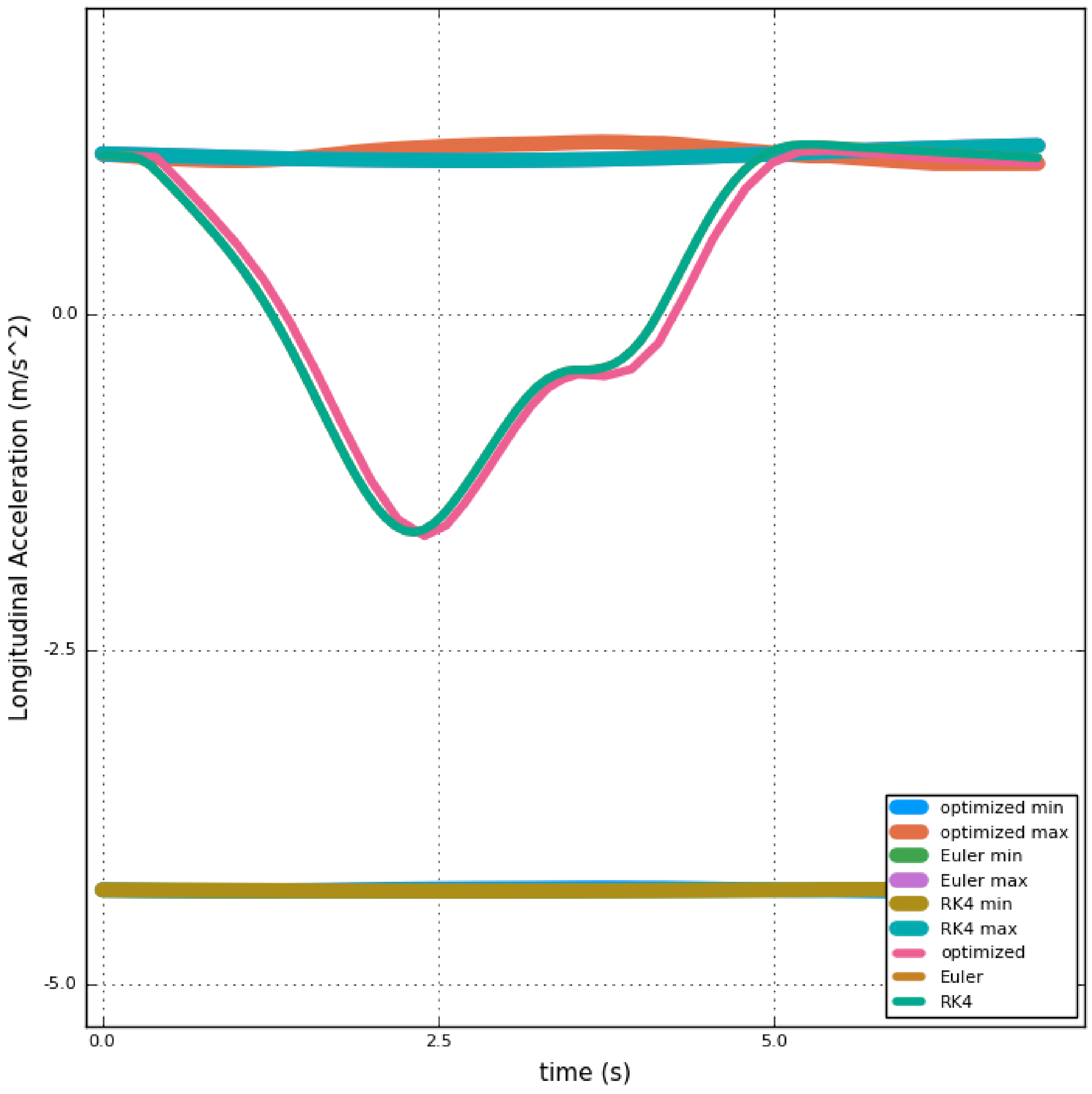
- The differences between the optimized and the actual (using the differential equation solvers) are acceptable.
- The animation simply demonstrates that the vehicles are avoiding obstacles:
  - The time scale is not correct, because there are different numbers of frames
  - It does not show the RK4 solution:
    - It is almost exactly the same as the Euler solution

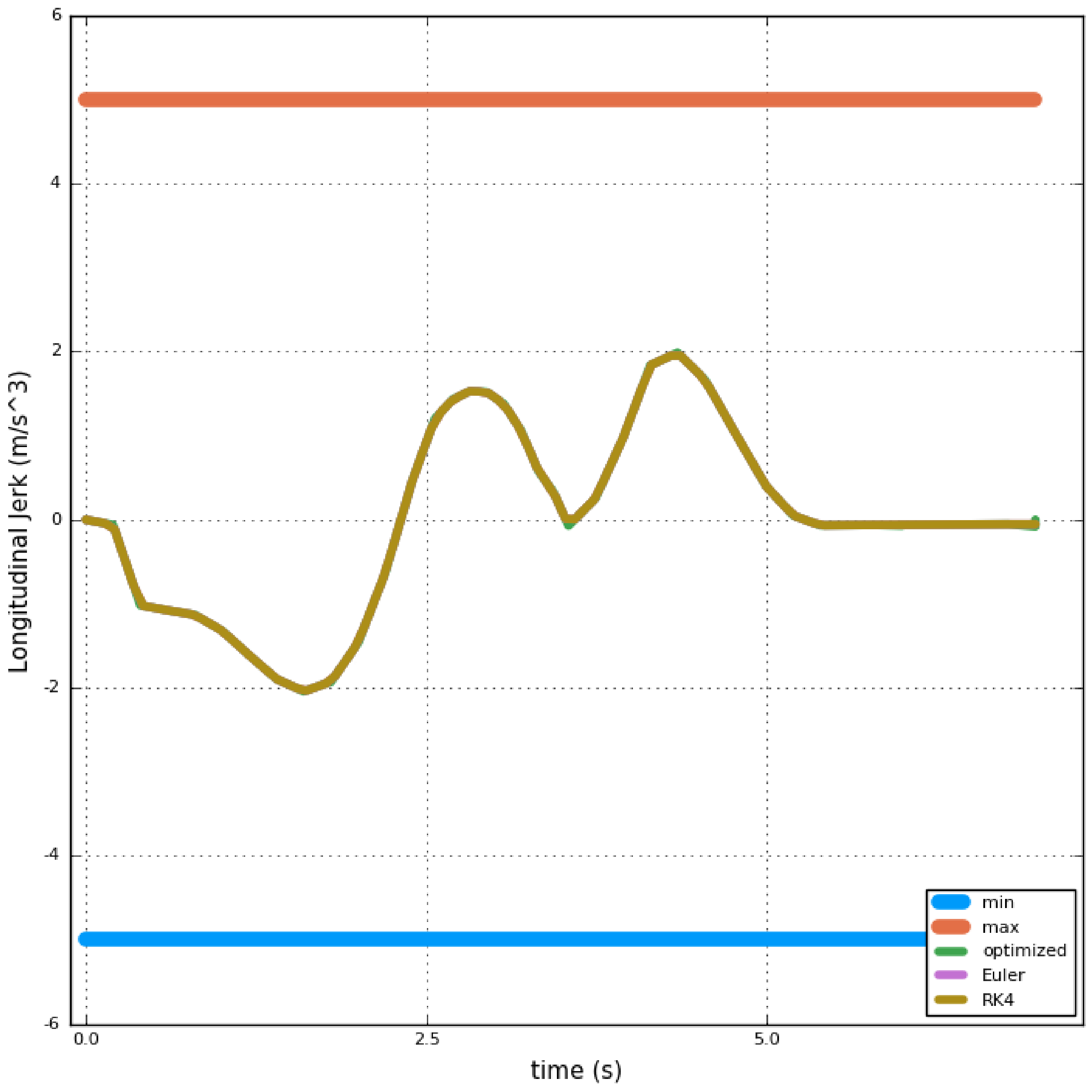
Taking a look at the states:

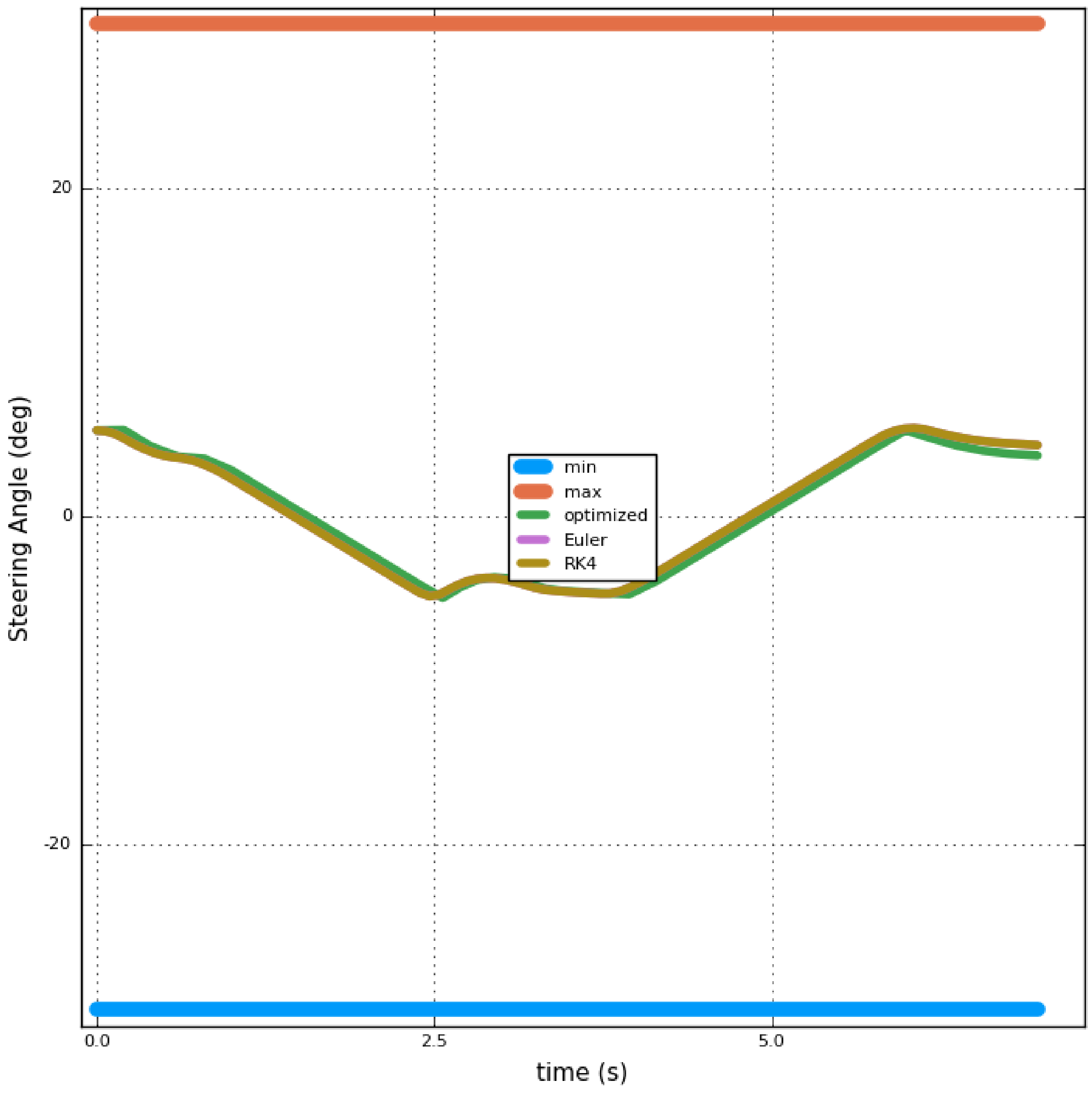


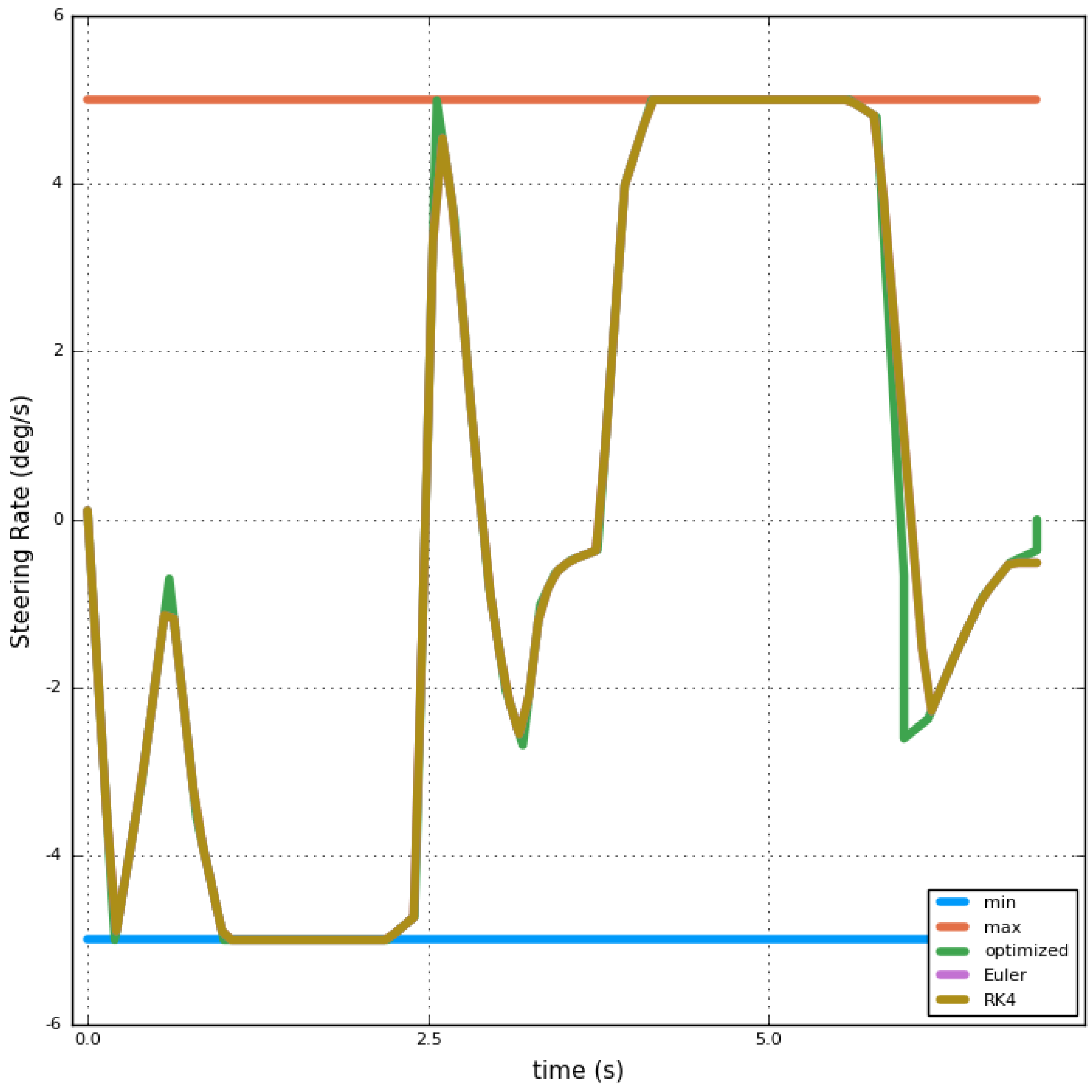
A closer look at the states and controls:

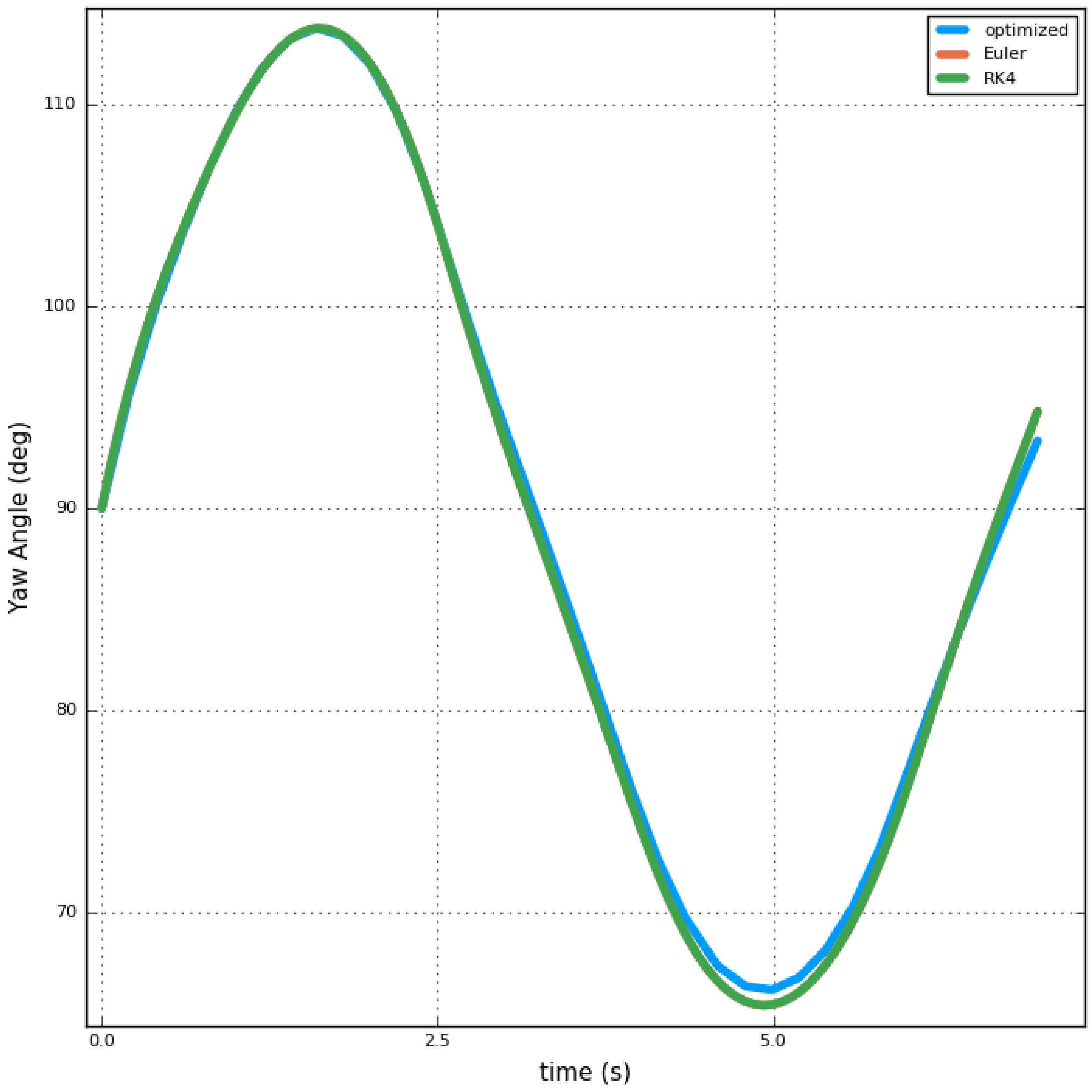

Longitudinal Velocity (m/s)
time (s)
30
20
10
0.0
2.5
5.0
min
max
optimized
Euler
RK4

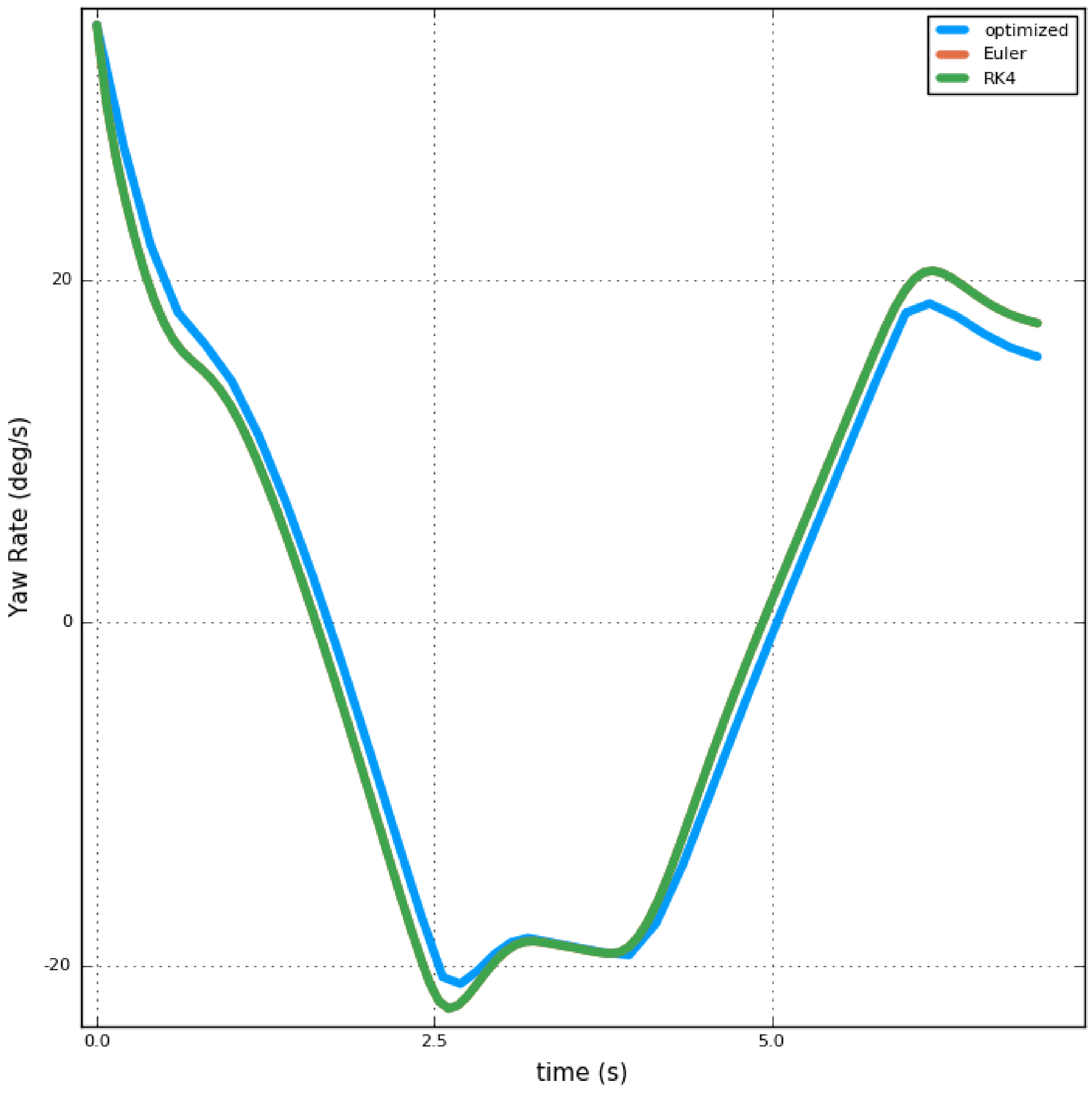

Longitudinal Acceleration (m/s^2)
0.0
-2.5
-5.0
0.0
2.5
5.0
time (s)
optimized min
optimized max
Euler min
Euler max
RK4 min
RK4 max
optimized
Euler
RK4

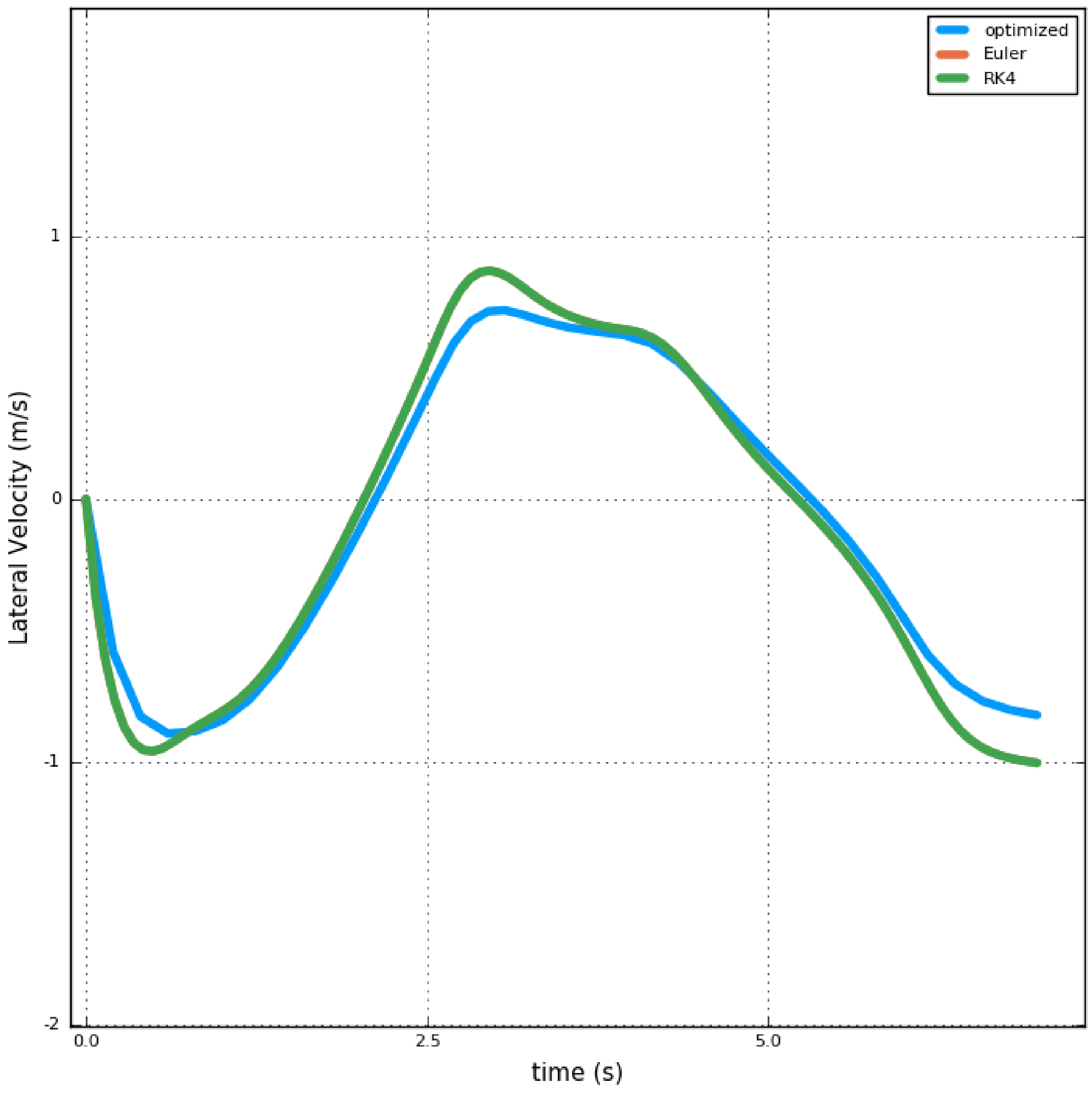

6
4
2
0
-2
-4
-6
Longitudinal Jerk (m/s^3)
0.0
2.5
5.0
time (s)
min
max
optimized
Euler
RK4

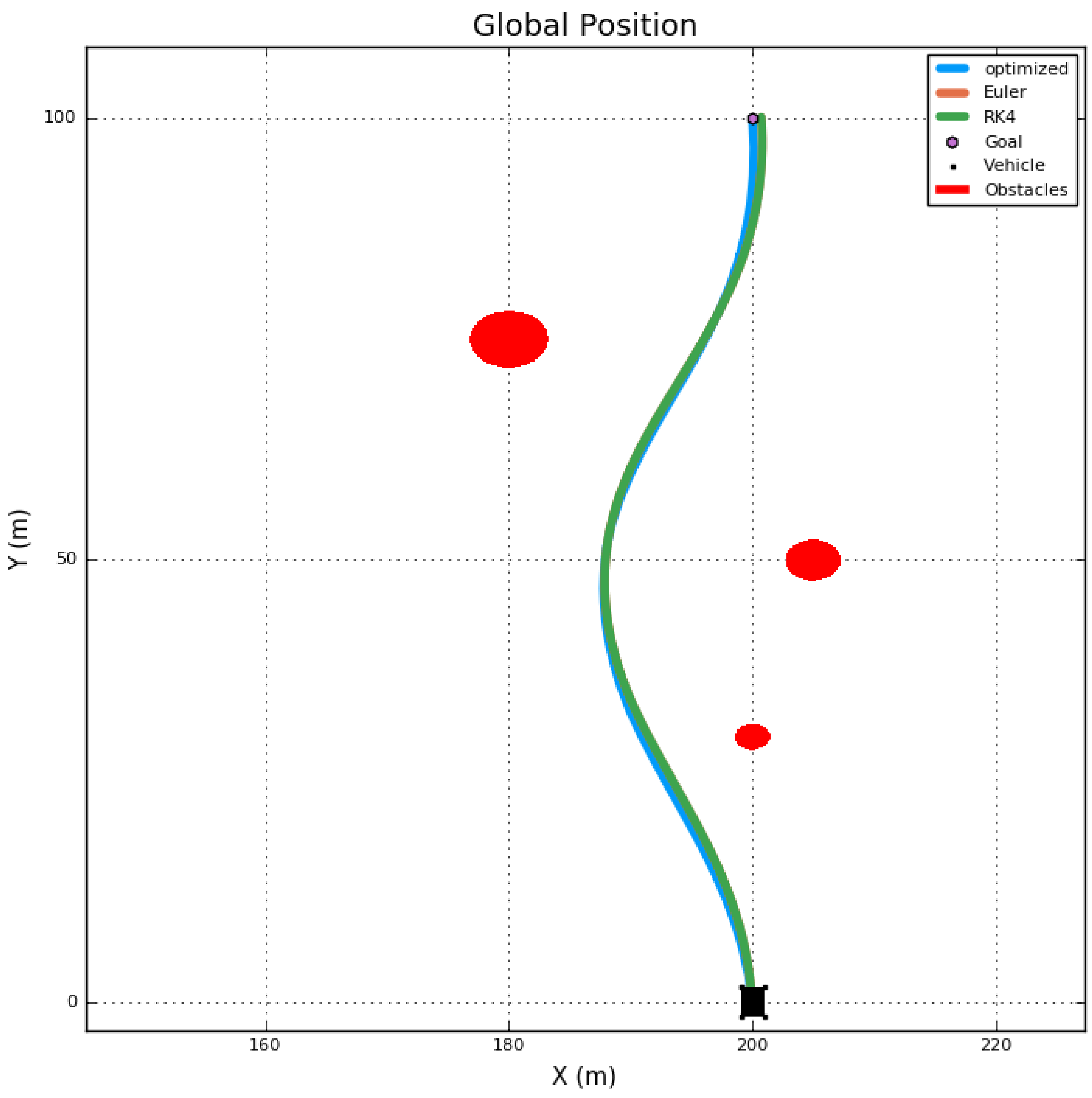

Steering Angle (deg)
20
0
-20
0.0
2.5
5.0
time (s)
min
max
optimized
Euler
RK4


6
4
2
0
-2
-4
-6
Steering Rate (deg/s)
0.0
2.5
5.0
time (s)
min
max
optimized
Euler
RK4

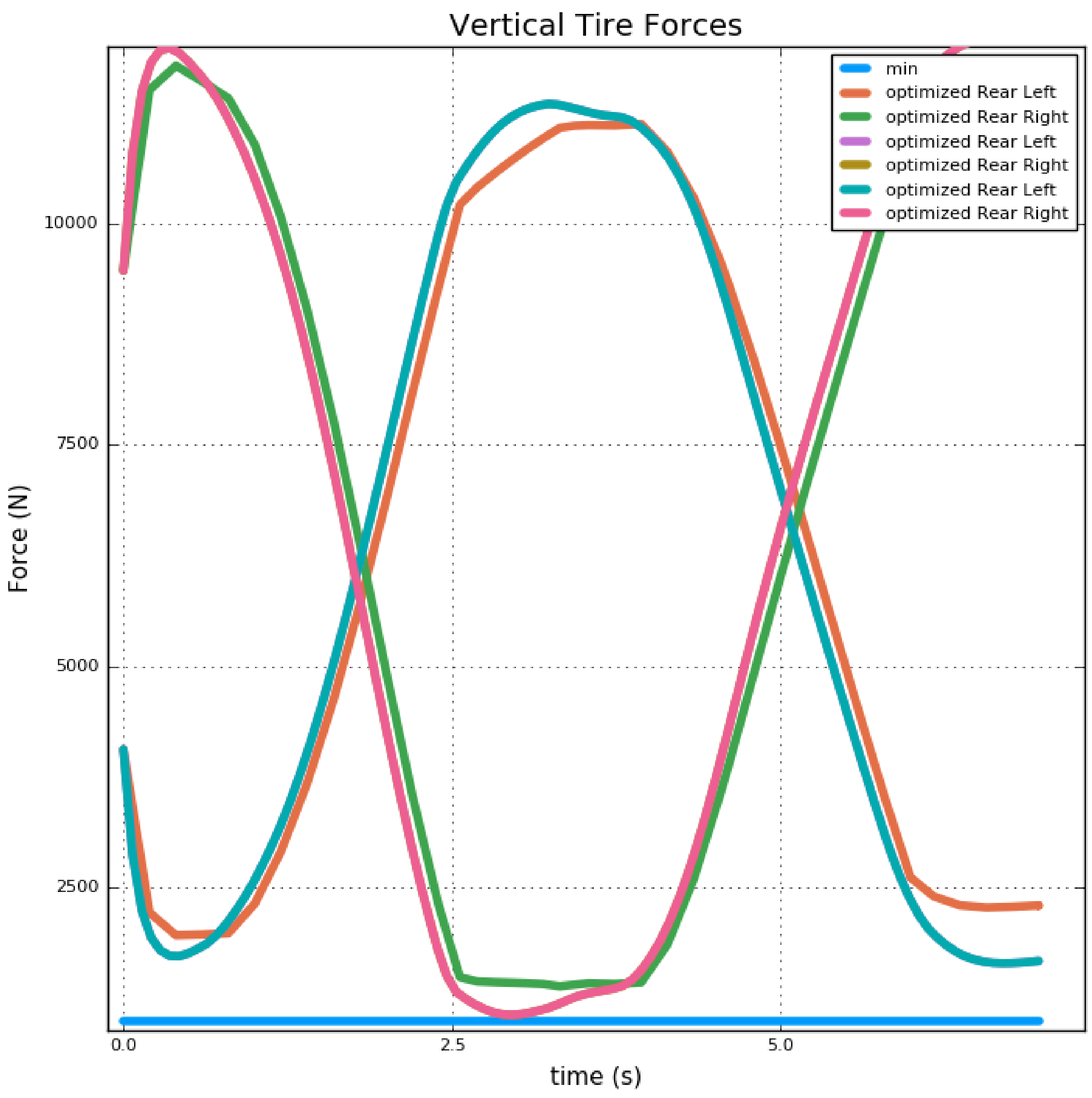

optimized
Euler
RK4
110
100
90
80
70
Yaw Angle (deg)
0.0
2.5
5.0
time (s)

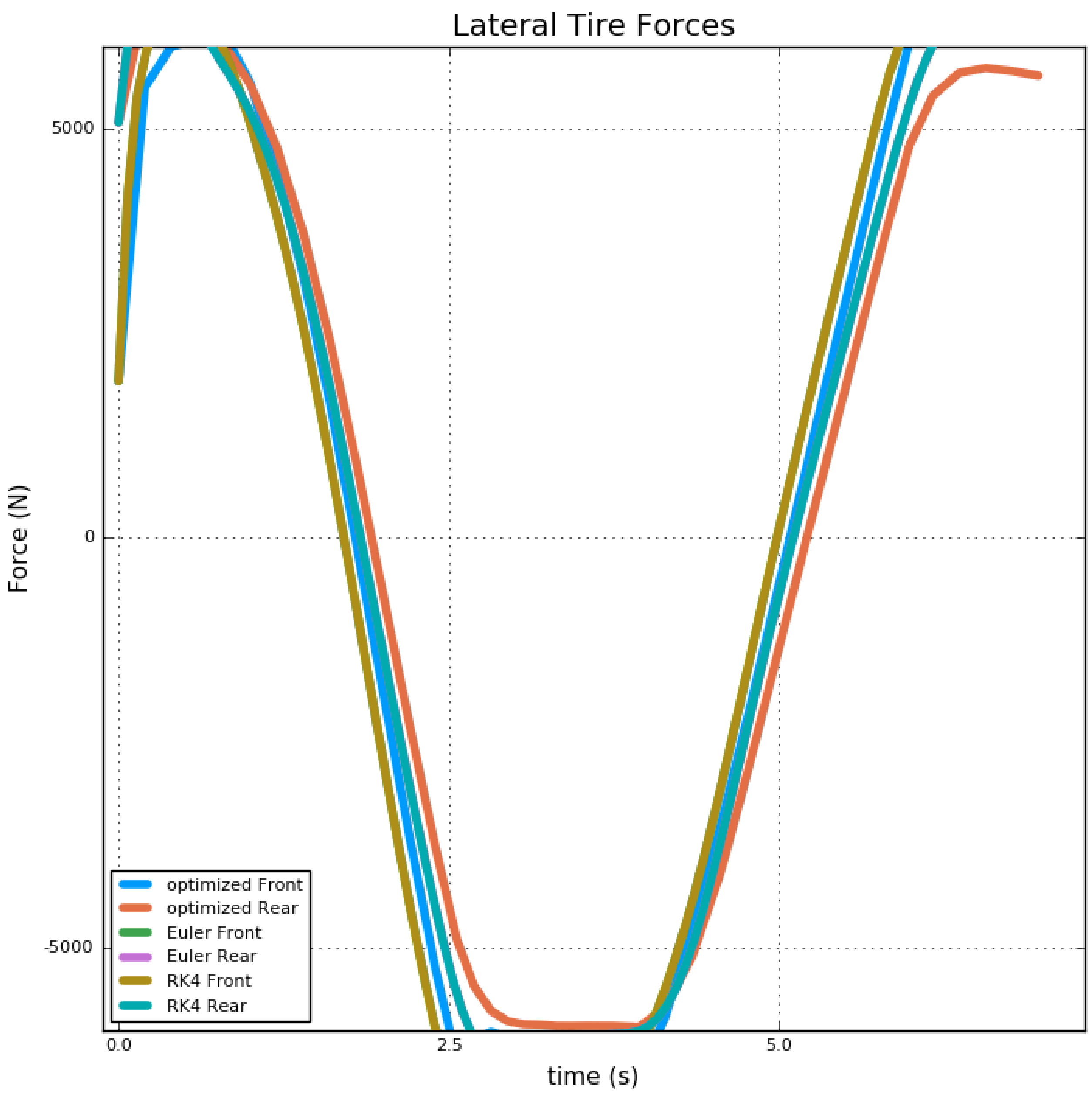

optimized
Euler
RK4
20
0
-20
Yaw Rate (deg/s)
0.0
2.5
5.0
time (s)


optimized
Euler
RK4
Lateral Velocity (m/s)
1
0
-1
-2
0.0
2.5
5.0
time (s)



The tire forces are:


Vertical Tire Forces
Force (N)
time (s)
10000
7500
5000
2500
0.0
2.5
5.0
min
optimized Rear Left
optimized Rear Right
optimized Rear Left
optimized Rear Right
optimized Rear Left
optimized Rear Right



5.1.2 NLOptControl.jl

This is a package that I am developing that allows for the easy inclusion of ordinary differential equation (ODE) constraints into the optimal control problem. Initially, I will focus on developing functionality to use a psuedospectral method to solve the optimal control problem.

- The package is aqui.
- The documentation for this package is here.

5.1.3 LiDAR.jl

This is a package that is being built to model a LiDAR sensor.

- The package is aqui.
- The documentation for this package is here.

5.1.4 Senarios.jl

This is a package that to represent various test cases.

Currently it is geared toward:

- Keeping good track of and documenting parameters

5.2 Moving Obstacle Avoidance

CHAPTER 6

Research Notes

6.1 LiDAR

6.2 What to study next?

CHAPTER 7

TO DO

7.1 Gantt Chart for Fall 2016 - Fall 2017

Acronymn	Description
OCP	optimal control problem
CL	close/d loop
OL	open loop
KE	known environment
UE	unknown environment
SO	static obstacles
MO	moving obstacles
VM	vehicle model
DOF	degree of freedom
VEXP	virtual experimentation
PEXP	physical experimentation

7.2 Current High Level Goals

1. Close the Loop with the 14 DOF vehicle model

- Put MATLAB on my UBUNTU machine
- Create interface between MATLAB and julia

2. Close the Loop with the 3 DOF vehicle model

- Create functionality in julia to solve ODEs given optimized control signals -> **DONE**
- Figure out what is going on with infeasibility issue -> **Current Focus**

7.3 Weekly Meeting | 1/6/2017

Accomplishments

1. Developed functionality for:

- Lagrange basis polynomials
- LGR multiple interval

- NLP Problem Initialization

2. Developed notes for:

- Lagrange basis polynomials
- LGR multiple interval
- NLP Problem Initialization

3. Developed examples for:

- Lagrange basis polynomials
  - Simple Interpolation
  - Investigated Runge's Phenomena
- LGR multiple interval
  - Calculate derivatives
  - Calculate integrals
  - Approximate state at the end of the mesh grid
- NLP Problem Initialization
  - Problem Initialization
  - Single Interval
  - Multiple Interval
  - Multiple States

Goals

1. Continue to on package NLOptControl.jl package
2. Meet with Jiayan

7.4 Future Ideas: