

Digital Twin for ^{40}Ca Trapped Ion Experiments
Lab Notebook: Jewoo Park
Supervised by: Koray Mentesoglu

Background Reading + Concepts

Concepts:

- Trapped ion control
- Programming in Julia

Resources

1. IonSim: https://examples.ionsim.org/tutorial-notebooks/getting_started.html

Tentative Timeline

Goal: Build a digital twin of our trapped ion experiment to simulate expected results

Week 1: Join lab, do paperwork/training

Weeks 2: Install Julia and IonSim, go through example notebooks, learn about trapped ion dynamics

Week 3,4: Implement qubit-only dynamics, noiseless.

Week 5,6: Add shot-to-shot noise

Week 7,8: Implement experiments with 1 ion and motional mode control

Week 9-13: Experimentally relevant exploration (to be decided)

Timeline

Week 1 (July 8 ~ July 15)

- Reading Hempel thesis and understanding the basic concept of Ion trap
 - Reference <https://examples.ionsim.org/intro.html>
 - Familiarize with IonSim and Julia language, initial background setup

Week 2 (July 16 ~ July 22)

- Identified a critical version conflict between the `IonSim.jl` package and its dependency, `Optim.jl`
 - Manually edited the package source file (`iontraps.jl`) to correct the behavior and proceed with simulations.
 - Focused mainly on implementing Rabi oscillations and applying laser pulses on a single Ca+ ion.

```
iontraps.jl X final_tutorial.ipynb Untitled-1.ipynb
C:\Users\jewoo>julia> packages > IonSim > s9QSU > src > iontraps.jl
459 function searchfor_trappingpotential_parameters(
460     )
461     @assert length(M) == length(Q) "length(M) != length(Q)"
462     maxM = maximum(M)
463     value = diagonalize_Kij(M, Q, k, maxM, 0, 0; optimize=true)
464     kz = (target_eigenfrequencies.z / value)^2 * maxM
465     lowerbounds = [-1, -1]
466     upperbounds = [Inf, Inf]
467     difference = 0
468     if target_eigenfrequencies.x != target_eigenfrequencies.y
469         initial = [10kz, kz / 2]
470     else
471         res = optimize(
472             x ->
473                 sequentially_diagonalize(M, Q, target_eigenfrequencies, kz, x[1], x[2]),
474             lowerbounds,
475             upperbounds,
476             initial,
477             Fminbox(WelderMead()),
478             #Optim.Options(g_abstol=1e-12, g_reltol=1e-6),
479             Optim.Options(g_abstol=1e-12),
480         )
481         xfreqs = [i[1] for i in diagonalize_Kij(M, Q, x, kz, res.minimizer...)]
482         yfreqs = [i[1] for i in diagonalize_Kij(M, Q, y, kz, res.minimizer...)]
483         difference += abs(target_eigenfrequencies.x - xfreqs[1])
484         difference += abs(target_eigenfrequencies.y - yfreqs[1])
485         res = [kz, res.minimizer...]
486     end
487     res = [kz, res.minimizer...]
488     res = optimize(
489         x -> sequentially_diagonalize(M, Q, target_eigenfrequencies, kz, x[1], 0),
490         lowerbounds[1:1],
491         upperbounds[1:1],
492         initial[1:1],
493         Fminbox(WelderMead()),
494         #Optim.Options(g_abstol=1e-12, g_reltol=1e-6),
495         Optim.Options(g_abstol=1e-12),
496     )
497     xfreqs = [i[1] for i in diagonalize_Kij(M, Q, x, kz, res.minimizer[1], 0)]
498     yfreqs = xfreqs
499     difference += abs(target_eigenfrequencies.x - xfreqs[1])
500     res = [kz, res.minimizer..., 0]
501     end
502     only_positive_eigenvalues = all(vcat(xfreqs, yfreqs) .> 0)
503     only_nonnegative_trapping_parameters = all(res .>= 0)
504     error_string = (
505         "Solver failed to find a normal mode structure compatible with 'characteristicfrequencies'."
506     )
507 
```

Jewoo Park 6:03 PM

Hey everyone,

As I mentioned to some of you earlier, I'm traveling this week and unfortunately won't be able to join our meetings. I just wanted to give a quick update on what I've been working on! Over the past few days, I've been reviewing the basic syntax of Julia and exploring IonSim in general. While experimenting with some of the functions IonSim provides, I ran into a strange issue with the linearchain function. After some digging, I discovered that the problem was due to a conflict between the Optim and IonSim packages.

To work around it, I manually edited the IonSim/iontraps.jl source file—commenting out the original function and modifying the parameters to achieve the desired behavior. Since Caleb and I both experienced this issue, and I've tried installing IonSim in various environments, I suspect this might be a common problem for new users. I've included a screenshot of the modified code below for reference.

For the rest of this week, I'll be focusing on implementing and optimizing the Pulse and Rabi functions we discussed last week.

I'll aim to share some results by next week. I'll also look into identifying compatible versions of IonSim, Optim, and Julia so we can avoid having to patch the source manually in the future.

Apologies again for missing the meetings—if there are any updates or things I should work on during the week, I'd really appreciate it if someone could let me know. I'll do my best to catch up over the weekend once I return.

Thanks again!

image.png ▾

Week 3 (July 23 ~ July 29)

- Investigated and found the correct version combination of all required packages: IonSim, Optim, QuantumOptics, and Julia.
- Dependency: Julia \Rightarrow Optim \Rightarrow QuantumOptics \Rightarrow IonSim
- Identified that Optim v1.12.0 removed a parameter required by IonSim. Downgrading Optim to v1.11.0 resolved the conflict.
- Shared the stable package environment with the team, providing a solution that avoids manual source code changes.
- Optim v1.12.0 #1135 deleted g_reltol parameter
- <https://github.com/JuliaNLsolvers/Optim.jl/pull/1135/files>
- <https://github.com/JuliaNLsolvers/Optim.jl/pull/1135>

```
(@v1.10) pkg> status
Status `C:\Users\jewoo\.julia\environments\v1.10\Project.toml'
^ [717857b8] DSP v0.7.10
[7073ff75] IJulia v1.29.0
[511e77fe] IonSim v0.5.1
^ [429524aa] Optim v1.11.0
[d330b81b] PyPlot v2.11.6
[6e0679c1] QuantumOptics v1.2.3
[295af30f] Revise v3.8.0
[789caeaf] StochasticDiffEq v6.80.0
Info Packages marked with ^ have new versions available and may be upgradable.

(@v1.10) pkg> |
```

Correct version of all the dependency package

Jewoo Park 11:12 AM
Optim v1.12.0 #1135 deleted g_reltol parameter
V1.11.0 should solve the error
[https://github.com/JuliaNLsolvers/Optim.jl/pull/1135](https://github.com/JuliaNLsolvers/Optim.jl/pull/1135/files)
I actually found the correct version that does have the correct g_reltol parameter. Optim updated their parameter for some of the optimization function I believe and it collides with IonSim since IonSim did not seem to take that update into account. If we downgrade Optim package to v1.11.0, I believe we would have fully working connection between Julia -> Optim -> IonSim
This is my package versions that do not need hard code change of the IonSim/Iontrap.jl file
image.png ▾

```
(@v1.10) pkg> status
Status `C:\Users\jewoo\.julia\environments\v1.10\Project.toml`
^ [717857b8] DSP v0.7.10
[7073ff75] IJulia v1.29.0
[511e77fe] IonSim v0.5.1
^ [429524aa] Optim v1.11.0
[d330b81b] PyPlot v2.11.6
[6e0679c1] QuantumOptics v1.2.3
[295af30f] Revise v3.8.0
[789caeaf] StochasticDiffEq v6.80.0
Info Packages marked with ^ have new versions available and may be upgradable.

(@v1.10) dep> |
```

C Caleb Walton 11:33 AM
^ this did fix it

C Caleb Walton 11:42 AM
what day are we supposed to meet @Sara Mouradian

Our Team's slack channel after the version share

Week 4 (July 30 ~ August 5)

- Implemented a laser intensity function to simulate phase shifts.
- Created a generalized `pulse` function to apply desired laser pulses to the ion, ensuring easy manipulation during simulations
- Observed and plotted the desired phase shift of the ion with various laser detunings.

```

function pulse(T:: Chamber , tspan, pitime)
    # Define the laser that will drive the transition
    L = T.lasers[1]

    # Combine all components into a single Trap object, which represents the full experiment
    # This is the main object that holds the entire state of our physical system.

    pi2_time = pitime*1e6/2

    res_intensity = intensity_from_pitime(L, pitime, T.iontrap.ions[1], ("g", "e"), T)

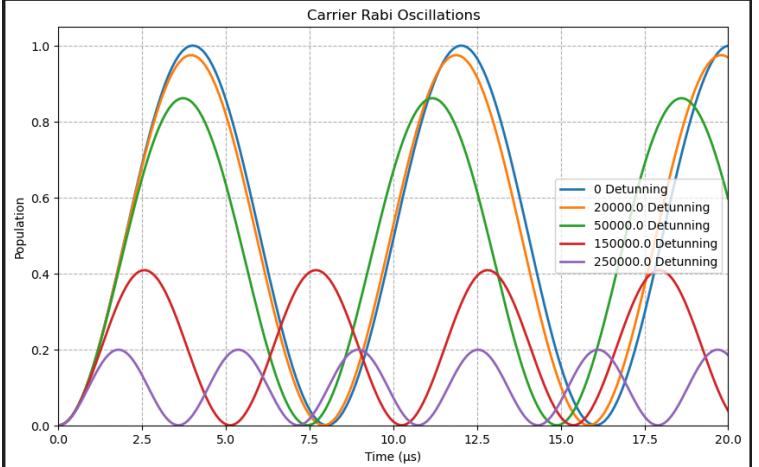
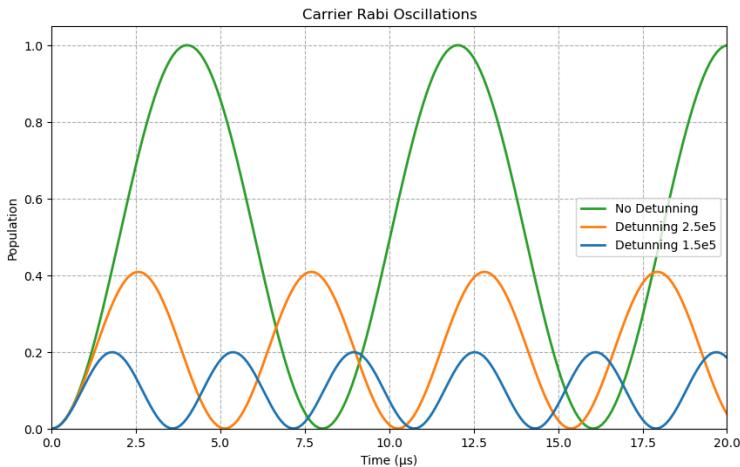
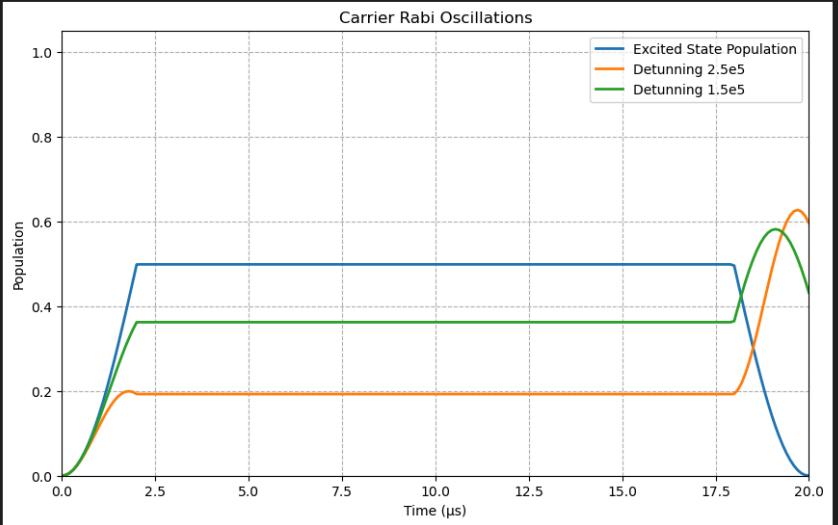
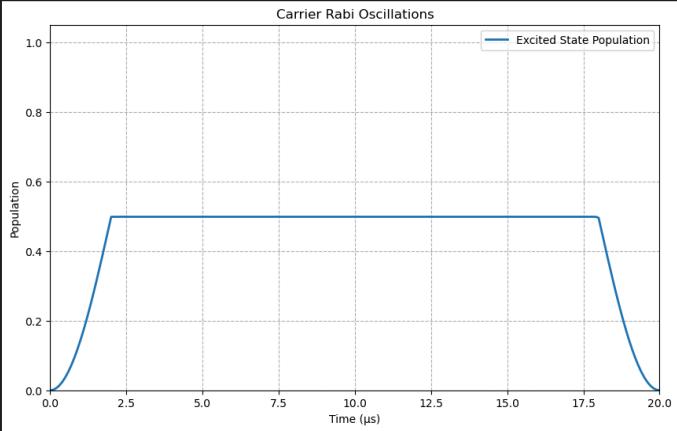
    function intensity_funtion(t)
        if(t<=pi2_time)
            return res_intensity
        elseif(t>=tspan[end] - pi2_time)
            return res_intensity
        else
            return 0.0
        end
    end

    intensity!(L, intensity_funtion)

    function phase_funtion(t)
        if(t<=pi2_time)
            return 2*pi
        elseif(t>=tspan[end] - pi2_time)
            return pi
        else
    end

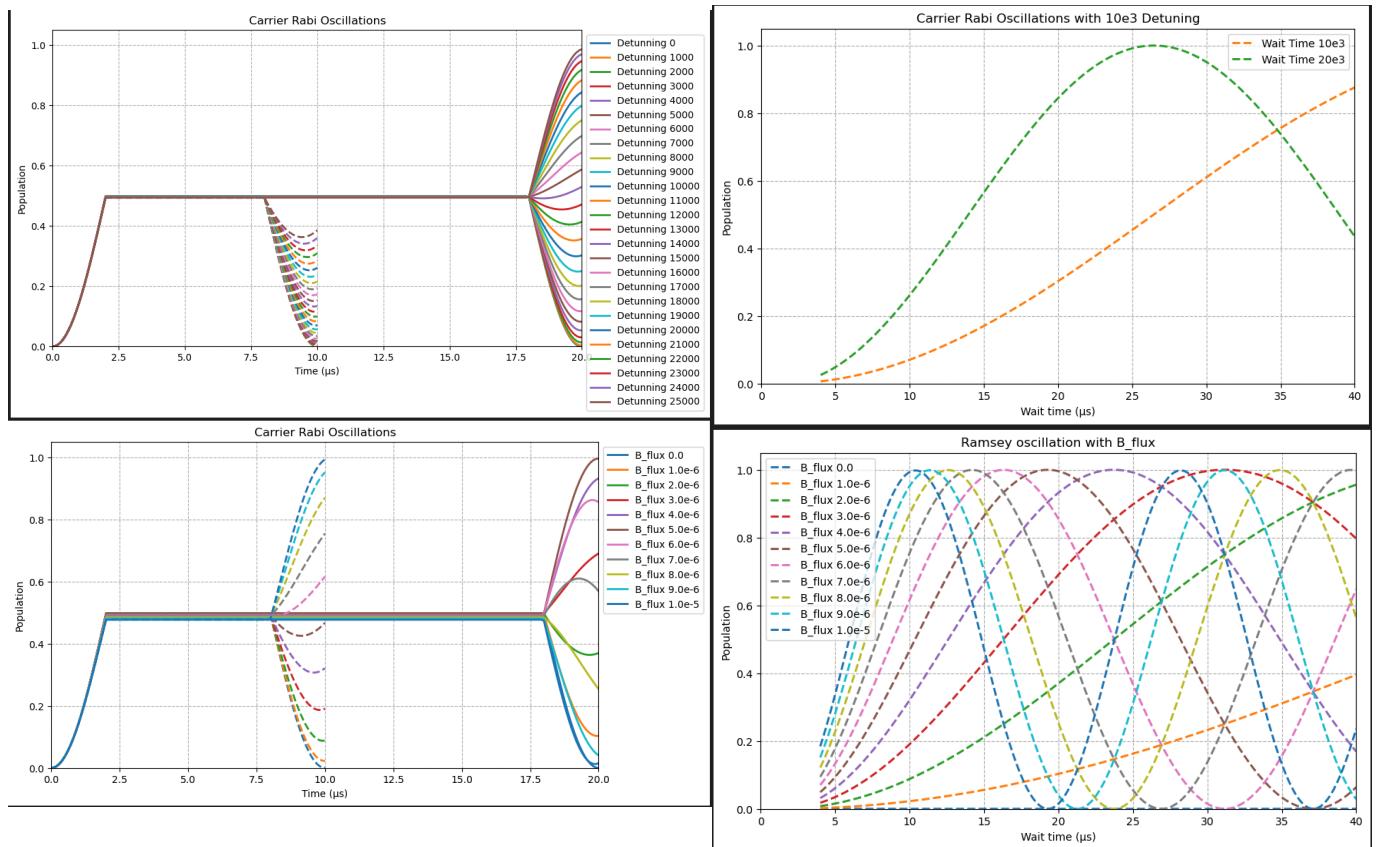
```

$\lambda: 7.291469968190796 \times 10^{-7} \text{ m}$
 $\Delta: 0 \text{ Hz}$
 $\vec{r}: (x=0.7071067811865475, y=0.0, z=-0.7071067811865475)$
 $\vec{k}(t=0): (x=0.7071067811865475, y=0.0, z=0.7071067811865475)$
 $I(t=0): 1.030939867952202 \times 10^6 \text{ W/m}^2$
 $\theta(t=0): 1.0 - 2\pi$



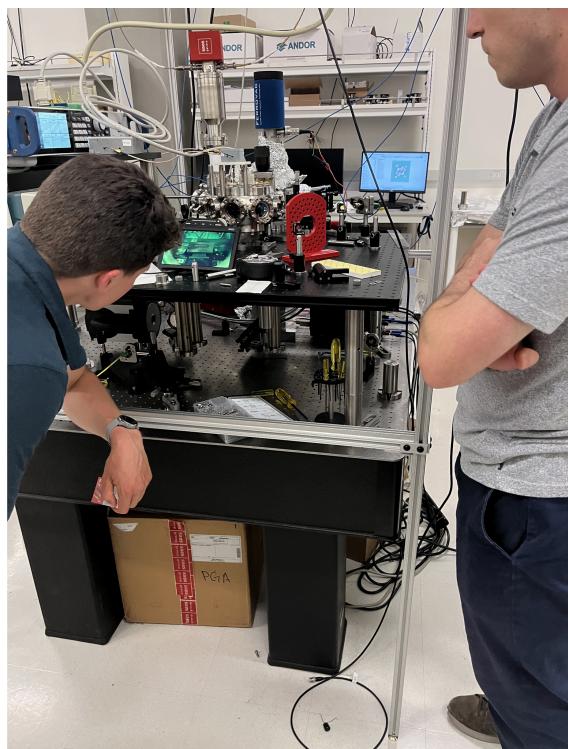
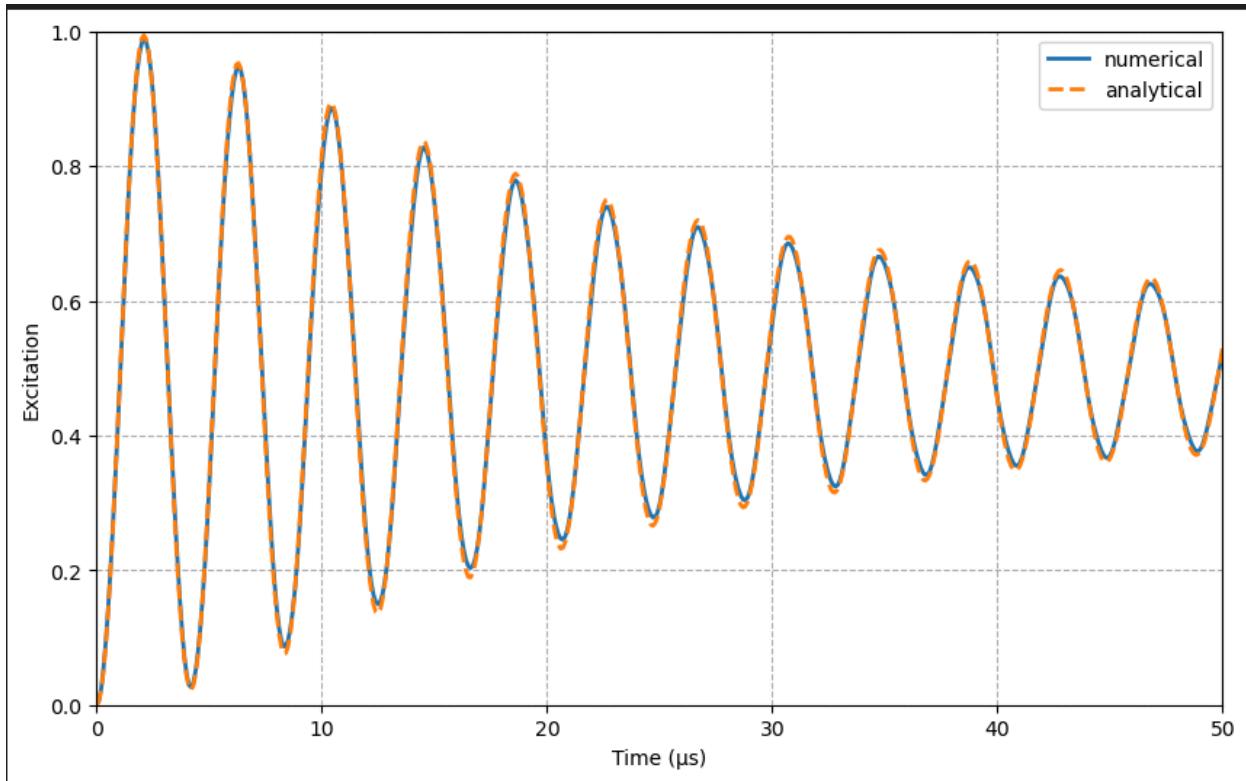
Week 5 (August 6 ~ August 12)

- Changed the observing parameter to "wait time" rather than a continuous time interval on the x-axis in order to better mimic the discrete, sequential nature of a real-world experiment.
- Plotted the population change with respect to wait time to simulate a Ramsey experiment.
- Pivoted focus to include magnetic flux shifts to observe similar phase shift behavior.
- Successfully observed Ramsey oscillations with given detunings in the magnetic field.



Week 6 (August 13 ~ August 19)

- Introduced a new degree of freedom in the single ion state by modeling it with a thermal state instead of a single Fock state.
- Observed the characteristic decay in Rabi oscillation amplitude due to the incoherent sum of oscillations from different initial Fock states.



Assisted with the alignment of laser and optical pathways for the fast chamber system

Summary & Future Work

Summary: Over the past six weeks, significant progress was made in developing a digital twin for the ${}^{40}\text{Ca}^+$ experiment. Key accomplishments include establishing a stable and reproducible simulation environment for the team by resolving critical package dependency issues in Julia. Core single-qubit operations were successfully implemented and verified, including Rabi oscillations, Ramsey interferometry, and the application of custom laser pulses. The model was further advanced by incorporating more realistic experimental factors, such as magnetic field effects, laser detuning, and initial thermal states.

Future Work: The immediate next steps will focus on expanding the simulation's complexity and realism. This includes implementing shot-to-shot noise to better model experimental uncertainty. The goal is to progress towards simulating specific experimental protocols relevant to the lab's current research.