QuTiP 튜토리얼(2020.02.26)

박찬후

목차

I. 개요

QuTiP은 양자 시스템의 계산과 시뮬레이션을 위한 Python 패키지이다. 우리 연구실에서는 NV를 처음 배운 학생들이 논문에서 공부한 내용을 직접 시뮬레이션 해보거나 실험해볼 내용을 미리 알아보는 등의 목적으로 사용중이다. 튜토리얼을 작성중인 필자 또한 동일한 이유로 QuTiP을 공부했으며 향후 같은 활동을 하게 될 학생들을 조금이나마 돕기 위해, 그리고 복습과 정리를 위해 본 문서를 작성한다. 필자는 컴퓨터 전공생이 아닐 뿐더러 QuTiP에 대한 지식도 아직 많이 부족하므로 문서의 내용을 항시 의심하면서 조심스럽게 진행하길 당부한다.

본 문서를 읽는 학생들의 프로그래밍에 대한 이해도를 수준별로 나눠보자면

1. 프로그래밍을 아예 해보지 않은 사람
2. 프로그래밍은 해봤지만 Python은 처음인 사람
3. Python을 해본 사람

정도일 것이다. 일단 1번에 속하는 사람은 거의 없을 것으로 보인다. 1번의 경우 QuTiP에 병행해서 프로그래밍, 특히 Python을 공부해야 하는데, Python에 대해서는 한국어든 영어든 많은 자료들을 인터넷상에서 찾을 수 있으므로 그것들을 참고하길 바란다. QuTiP 공부를 하면서 그때그때 필요한 것을 익혀나간다고 생각해도 좋을 것 같다. 본 문서의 내용은 2, 3번 정도를 가정하고 진행할 것이다.

본 문서의 많은 내용이 QuTiP documentation[1]을 참고하여 진행될 것이니 즐겨찾기 해두고 모르는 것이 나올 때마다 찾아보기를 **강력히** 권장한다. 또한 설치나 실행 과정에서 발생하는 다양한 문제들은 대부분 구글링을 통해 해결할 수 있으므로 잘 모르겠으면 구글에 검색한다는 마음가짐을 늘 지니길 바란다. 또한 본 문서의 내용은 **작성 시점에서의 최신 버전(20년 2월, QuTiP 4.5)을 기준으로** 하고 있으므로 사용자의 QuTiP버전에 따라 내용이 맞지 않을 수도 있다.

---

[1] http://qutip.org/docs/latest/index.html

II. QuTiP 설치

QuTiP은 Python 패키지이므로 우선 Python을 사용할 수 있는 환경이 필요하다. 여기서 설명하는 내용은 QuTiP documentation의 설치 안내[2]를 기준으로 하고 있으므로 설치 과정에서 필히 해당 문서를 함께 참고하자. 이 뒤부터는 설치 과정에서 필요한 명령어가 계속 제시되는데, 만약을 위해 명령어를 보고 바로 입력해 버리지 말고 좀 귀찮더라도 앞뒤 단락을 읽고 그 의미를 충분히 이해한 후 진행하길 바란다.

1. Google Colab

첫 번째로는 Google Colab[3]을 이용하는 방법이 있다. Colab은 구글의 하드웨어를 클라우드를 통해 사용할 수 있도록 해둔 것으로 다른 방법에 비해 하드웨어의 제약이 덜하고 컴퓨터에 Python을 설치하는 귀찮음이 없다. Colab의 기본적인 사용법 역시 머신러닝 공부하는 사람들이 작성해둔 가이드가 웹상에 널려있으니 그쪽을 참고하길 바라고, 본 문서에서는 다루지 않겠다. Colab에서 QuTiP을 사용하려면 **import 하기에 앞서 다음 명령어를 입력하여 QuTiP을 설치**해줘야 한다.

```
!pip install qutip
```

이후 QuTiP을 사용함에 따라 Colab에 기본적으로 설치되지 않은 패키지에 의해 오류가 날 수 있는데, 그때그때 출력되는 오류 메시지에 맞추어 필요한 패키지를 설치하면서 사용하면 될 것 같다. 이 방법의 단점은 매번 QuTiP을 새로 설치해주어야 한다는 것이다. 그렇게 시간이 오래 걸리지는 않지만 귀찮은 일이다. 필자는 Colab을 사용하여 QuTiP을 제대로 사용해본 적이 없는 관계로 이 방법에 대한 설명은 이정도로 하자.

---

[2] http://qutip.org/docs/latest/installation.html

[3] https://colab.research.google.com/

2. Anaconda

다음 방법이자 권장하는 방법은 **Anaconda를 이용해서 컴퓨터에 Python 환경을 만들고 QuTiP을 설치하는 것**이다. 설치에 난항이 있을 수 있으나 한 번 해두면 이후에는 사용만 하면 된다는 장점이 있다. 가장 먼저 해야할 것은 **Anaconda를 설치**하는 것이다. 공식 홈페이지의 다운로드 페이지[4]에서 자신의 운영 체제에 맞는 Python 3.7 버전의 최신 버전 Anaconda installer를 다운로드 하여 설치하면 된다. 설치를 완료하면 다음 단계로 넘어가자. 우선 최대한 간략화한 설치 과정은 다음과 같다. 자세한 설명은 그 아래를 참고하자.

1. Conda 환경 만들기

   conda create -n qutip-env python=3

   Proceed ([y]/n)? 에 y를 입력

2. Conda 환경 활성화하기

   conda activate qutip-env

3. 필요한 패키지 설치하기

   conda install numpy scipy cython matplotlib pytest pytest-cov spyder

4. Visual Studio 2015 설치하기

   https://visualstudio.microsoft.com/ko/vs/older-downloads/

   Visual Studio Dev Essentials에 가입해야함

   C++ 컴파일러 설치 옵션 반드시 선택

5. Conda-forge 채널 추가

   conda config --append channels conda-forge

6. QuTiP 설치

   conda install qutip

[4] https://www.anaconda.com/distribution/

7. Spyder 실행

(conda activate qutip-env)

spyder

8. QuTiP 테스트(Spyder에서)

import qutip.testing as qt

qt.run()

여기서부터 두 가지 중 하나를 선택할 수 있다. 하나는 **기본 Conda 환경에 Qutip을 설치**하는 것이고, 다른 하나는 **QuTiP 전용으로 새 환경을 만들어 사용**하는 것이다. Conda environment는 간단히 말해 패키지를 깔아서 사용할 수 있는 독립된 공간이다. 각각의 환경 사이에서는 설치된 패키지가 공유되지 않으며 당연히 동일한 패키지가 환경별로 각각 깔릴 수도 있다. 이런 것이 필요한 이유는 사용해야 하는 패키지에 따라 필요로 하는 다른 패키지의 종류와 버전 등이 다를 수 있고, 이 경우 패키지의 버전 차이에 따른 충돌이 발생하기 쉽기 때문이다.

새 환경을 만드는 것은 처음 하는 사람에겐 좀 헷갈리고 시간이 더 걸리지만 여러 장점이 있다. 만일 무언가 알 수 없는 문제가 생긴다면 해당 환경을 삭제해 버리고 재설치 하는 등의 해결책을 사용할 수도 있고, 앞서 말했듯 특정 상황에 필요한 패키지만 설치해 사용할 수 있으므로 버전 충돌 문제가 발생할 확률이 매우 낮다. 후자의 경우 QuTiP 이외의 목적으로 Python을 사용하지 않는다면 별로 중요하진 않지만, 우리는 사람 일은 어떻게 될지 모르며 Python은 여러가지 분야에서 사용된다는 사실을 기억해야 한다. 따라서 필자는 새 환경을 만드는 것을 권하고 싶다. 기존 환경에 그대로 설치하고자 한다면 2-ii로 바로 넘어가면 된다.

2-i. Conda 환경 만들기

우선 'qutip-env'라는 이름의 새 환경을 만든다고 가정하자. 이름은 자유롭게 정할 수 있지만 너무 길고 헷갈리는 이름은 좋지 않다. 다른 이름을 쓰고 싶다면 이후 나오는 명령어들에서 qutip-env라고 되어있는 부분을 자신이 원하는 이름으

로 바꿔줘야 한다. 환경을 만들기 위해서는 윈도우 사용자라면 명령 프롬프트, 맥이나 리눅스 사용자라면 터미널을 열고 다음 명령어를 입력해주면 된다.

conda create -n qutip-env python=3

그림 1. 환경 만들기. 여기서는 환경 이름으로 'qutip'을 사용했다.

Proceed ([y]/n)? 에 y를 입력해주면 설치가 진행되고 잠시 기다리면 끝이다. 만들어진 환경을 이용하려면 설치 완료 후에 나오는 안내대로 다음의 명령어를 입력해줘야 한다.

conda activate qutip-env

환경을 활성화하면 명령창 좌측에 환경 이름이 표기되고, 이후 실행하는 명령어는 해당 환경을 기준으로 실행된다. 활성화된 환경을 비활성화하려면,

conda deactivate

를 입력하면 된다. 존재하는 환경의 목록을 확인하고 싶다면

conda env list

를 입력하면 된다. 이들 명령어를 기억하고 있으면 만들어둔 환경들 사이를자유롭게 이동할 수 있다. Conda environment에 대한 더 자세한 설명은 Anaconda userguide의 해당 항목5을 참조하길 바란다. **이제부터 아래 설명은 모두 새로 만든 환경이 활성화된 상태를 기준으로 한다.** 다음 단계로 넘어가기 전에 반드시 conda activate qutip-env를 입력해야 한다. 중간에 모종의 이유로 명령 프롬프트를 다시 열었다면 환경을 다시 활성화해야 한다는 것을 꼭 기억하길 바란다.

2-ii. QuTiP 설치 준비하기

환경이 준비가 되었다면 먼저 **QuTiP이 필요로 하는 다른 패키지들을 설치**해야 한다. 기본 환경에 QuTiP을 설치하고 있다면 이 과정을 생략해도 괜찮을 것이다. QuTiP 유저가이드에서는 세 가지 설치 목록을 제시하고 있고, 필자가 권하는 설치 목록은 다음과 같다.

conda install numpy scipy cython matplotlib pytest pytest-cov spyder

conda install은 뒤에 나열되는 패키지를 설치하는 명령어이다. 위 명령어는 유저가이드에 있는 것과는 조금 다르다.

pytest는 QuTiP의 작동과는 직접적으로 관련이 없으나 설치 후 작동 테스트를 하기 위해 필요하다. 또한 Jupyter Notebook을 제외하였는데, 사용하고 싶다면 위 목록에 jupyter notebook을 추가하여 실행하면 된다.

QuTiP documentation에 따르면 윈도우 사용자의 경우 QuTiP의 특정 기능을 사용하기 **위해선 Visual Studio 2015를 설치**해야 한다. VS는 용량이 수 GB 정도로 크고 설치에 시간도 꽤 걸리므로 컴퓨터에 용량이 부족하고 도전정신을 가진 학생이라면 설치하지 않고 넘어가보는 것도 가능하다. Documentation의 안내대로

[5] https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html

설치하고자 하는 학생은 다음 링크[6]에서 VS 2015 community edition을 무료로 다운로드 할 수 있고(단, 설치 파일 다운로드를 위해서는 Visual Studio Dev Essentials에 가입해야 한다. 다행히도 무료 서비스이다), 설치시에는 반드시 **C++ 컴파일러 설치 옵션을 선택**해야 한다. 이 링크[7]에서 논의된 내용에 따르면 VS를 설치하지 않고도 QuTiP을 설치하여 사용할 수는 있는 것으로 보인다.

2-iii. QuTiP 설치하기

필요한 패키지를 모두 설치했다면 마지막으로 QuTiP을 설치할 차례이다. 우선 **conda-forge 채널을 추가**해야 한다. Conda channel은, 말하자면 conda install로 설치하는 패키지가 저장된 장소이고 conda-forge는 그러한 채널 중 하나이다.

```
conda config --append channels conda-forge
```

위 명령어를 입력하면 conda-forge 채널을 채널 목록 최하위로 추가할 수 있다. 이제 모든 준비가 끝났으므로 QuTiP을 설치하면 된다.

```
conda install qutip
```

만일 이 명령어를 통한 설치가 실패했다면, 대신

```
pip install qutip
```

을 시도해볼 수 있다. PIP는 conda와 유사한 역할을 하는 Python 패키지를 관리해주는 시스템인데, Anaconda에서는 PIP를 사용한 후에 Conda로 패키지를 설치할 경우 문제가 발생할 수 있음을 경고하고 있다. 따라서 conda install을 우선시하되, 어쩔 수 없을 경우 conda install로 필요한 패키지를 가능한 한 모두 설치하고 마지막에 PIP를 쓸 것이 권장된다.

모든 설치 과정이 끝났다면 **설치가 잘 되었는지를 확인**해 보자. 먼저 QuTiP을 설치한 환경을 활성화하고 Spyder를 실행하자. 두 가지 방법이 있다. 하나는

---

[6] https://visualstudio.microsoft.com/ko/vs/older-downloads/

[7] https://github.com/qutip/qutip/issues/826

Anaconda Navigator를 실행한 다음 환경을 바꾸어서(그림 2 참조) Spyder 항목의 launch 버튼을 클릭하는 것이고, 다른 방법은 **명령창에서 환경을 활성화하고** spyder**를 입력**하는 것이다. Spyder가 열렸으면 새 파일을 만들고 아래의 코드를 입력한 후 화면 상단의 녹색 삼각형 버튼이나 F5 키를 눌러 실행시킨다. 최초 실행시 파일 저장 위치를 묻는 팝업창이 나오므로 원하는 위치에 적절히 저장하자.

```
import qutip.testing as qt
qt.run()
```

코드를 실행시키면 자동으로 테스트가 실행된다. 단, 설치 과정에서 pytest를 설치하지 않았다면 실행에 실패할 것이다. 이 때 테스트가 중간 어느 곳에서 한참을 기다려도 넘어가지 못하는 경우가 발생할 수 있는데, 그럴 경우 콘솔창 우상단의 빨간 네모 버튼을 눌러 중지시키도록 하자. 이것 외에도 여러 에러 메시지가 발생할 수 있다. 필자의 경험상 이런 문제들이 발생하더라도 마땅한 해결책을 찾기 힘들고 실제 QuTiP을 사용할 때 체감되는 영향이 없다. 신경쓰이겠지만 일단은 무시하고 넘어가도록 하자. **설치가 끝나면 이후부터는 다음 명령어를 통해 QuTiP을 import 하여 사용하면 된다.**

```
from qutip import *
```

튜토리얼 작성 도중에 QuTiP이 약 5개월만의 업데이트를 받았다. 기존 버전(4.4)에서는 3.1.0 버전 이상의 matplotlib 패키지와 호환이 되지 않았는데 이 버전부터 해당 오류가 해결되었다. 그럴 확률은 낮지만 만약 **자신이 설치한 QuTiP의 버전이 4.4 이하라면 matplotlib의 버전을 3.0.3으로 다운그레이드 하길 바란다.**

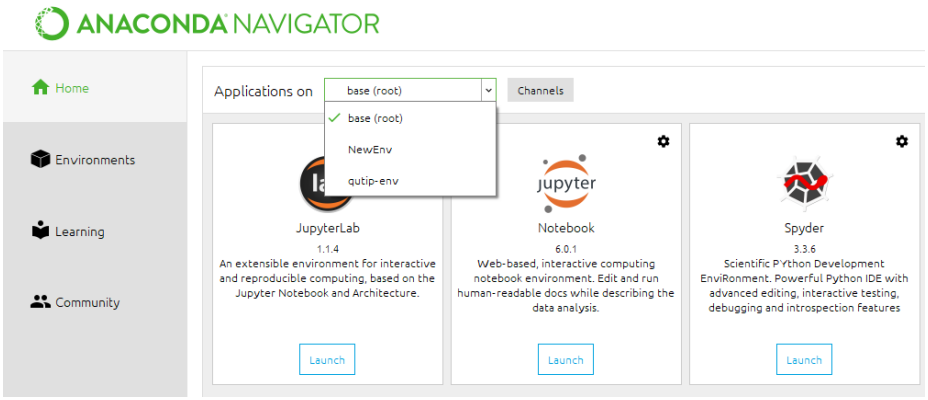

그림 2. Anaconda Navigator에서 환경 선택하기.

III. QuTiP 공식 튜토리얼

QuTiP은 documentation에 더해서 공식적으로 튜토리얼[8]을 제공하고 있다. 해당 튜토리얼은 파이썬에 대한 소개부터 시작해서 QuTiP의 기초와 여러 가지 적용 예시를 포함하고 있다. 따라서 QuTiP의 기능을 익히기 위해서는 우선 **제공되는 tutorial을 따라해 본 후 자신이 원하는 시스템에 적용해 보는 방식**으로 진행하는 것이 좋을 것이다. 해당 내용은 제법 양이 되는 바 본 문서에서 하나하나 번역하여 싣지는 않을 것이고, 몇 가지 도움이 될만한 것들을 소개하는 정도로 하겠다. 시작하기에 앞서 공통적으로 적용되는 내용을 말하자면 QuTiP을 사용하면서 계속해서 NumPy와 matplotlib을 사용하게 되므로 이들을 잘 알고 있는 것이 도움이 많이 된다. 이들은 Python에서 일반적으로 사용하는 패키지이므로 웹상에서 간단히 자료를 찾을 수 있다. 구글링을 생활화하자.

1. Introduction to QuTiP

   이름 그대로 QuTiP과 관련된 기본적인 내용을 소개하고 있다.

2. Groundstates: Jaynes-Cummings model in the ultrastrong coupling regime

   QuTiP을 어떤 식으로 사용하는지 볼 수 있는 예제이다.

3. Visualization demos

   QuTiP에서 사용하는 여러 가지 시각화 방식들을 보여준다.

4. Energy-level diagrams

   에너지 레벨 다이어그램을 그리는 방법.

5. Bloch-sphere animation

   제목은 애니메이션이라고 되어 있는데, 실질적으로는 Bloch Sphere를 이용한 표

[8] http://qutip.org/tutorials.html

현을 어떻게 할지에 대한 내용으로 봐도 좋다. 여러 이미지를 만들고 그걸 영상으로 만드는 방식이지만 필자의 경우 버전 문제로 제대로 작동하질 않아서 이미지 파일을 따로 저장하고 gif를 만들어주는 사이트[9]를 이용했다.

6. Master equation solver: Qubit dynamics
7. Master equation solver: Vacuum Rabi oscillations

6, 7은 Solver의 사용법과 시각화 방법을 보여주는 예제이다.

상기 목록은 필자가 직접 해본 것들 위주로 임의로 고른 것이므로 그 이외의 튜토리얼도 자유롭게 해보길 권장한다. 또한 개요 부분에서 언급했듯 documentation을 읽어보는 것이 많은 도움이 되므로 따라해보다가 **역할을 잘 모르겠는 함수 등이 있다면 QuTiP / NumPy / matplotlib 등의 documentation에서 꼭 설명을 읽어보도록 하자.** 이렇게 하면 튜토리얼보다 다루지 않은 옵션이나 활용법에 대해 더 잘 알 수 있게 되기도 한다.

또한 튜토리얼을 진행하면서 사소한 에러가 발생할 수 있는데 이런 것들은 이미 말했듯이 구글링으로 해결할 수 있는 경우가 대부분이다. Bloch sphere 관련 예제에서 그림이 나오지 못하고 에러메시지만 나올 경우 설치 과정에서 언급한 matplotlib 버전 문제일 가능성이 높다. 해당 내용에 각주로 달아둔 링크에 들어가서 에러 메시지가 같은 종류인지 확인하고 만약 그렇다면 matplotlib의 버전을 3.0.3으로 맞추길 바란다.

[9] 필자는 https://ezgif.com/maker 를 이용했고, 이외에도 많은 사이트가 있다.

IV. QuTiP을 이용한 2 level system 시뮬레이션

앞 섹션의 튜토리얼들을 모두 마쳤다면 이제 QuTiP의 사용에 어느정도 익숙해졌을 것이다. 다음으로 해볼 것은 실제 시스템을 시뮬레이션 해보는 것이다. 간단한 것부터 시작해보자. 각 부분의 말미에 필자가 작성한 코드를 첨부할 것이지만 **가능하면 주어진 정보만 가지고 스스로 만들어보도록 하자.** 팁을 주자면, QuTiP의 Qobj 오브젝트의 경우 print(qobj1)를 하면 해당 오브젝트의 정보를 보여주므로 그것을 어떤 식으로 사용할 수 있을지 생각하는 데 도움이 된다. 이하 수식들에서는 작성 편의상 $\hbar = 1$로 두고 작성하였다.

1. Zeeman splitting

Zeeman splitting은 간단히 외부 자기장에 의해 에너지 차이가 발생하는 현상이다. **2 level system에서 Zeeman splitting이 일어나는 것을 QuTiP으로 표현해 보자.** 다음 해밀토니안은 이러한 상황을 나타내고 있다.

$$H = \gamma B \sigma_z = \gamma B \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

이 해밀토니안의 eigenstate는 $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$그리고 $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$이며 이들의 energy eigenvalue는 $2\gamma B$만큼 차이난다. γ는 자기장에 비례해 splitting이 얼마나 발생하는지를 정하는 상수이고 B는 자기장의 크기이다. Zeeman splitting을 시각화하려면 이 **해밀토니안을 정의한 후 eigenvalue를 구하고 저장해서 그리면 된다.** 이런 상황에 사용할 수 있는 함수로는 eigenstates()가 있다. 이 함수는 입력한 해밀토니안에 맞는 eigenvalue와 eigenvector를 출력해주는 함수이다. 예를 들어, H라는 이름의 해밀토니안을 정의했다면 다음과 같은 코드를 사용해 eigenvalue, eigenvector를 얻을 수 있다.

```
evals, ekets = H.eigenstates()
```

QuTiP 튜토리얼에서 익힌 것들과 eigenstates()를 이용해서 그림 3과 같은 이미지를 얻어보자. 그래프의 색이나 범례 등을 넣을 수 있다면 좋겠지만 중요하지 않고, 일단은 두 energy level이 멀어지는 것을 나타내는걸 우선시하자. γ는 2.8

(MHz/G)의 값을 사용했다. 그림을 그리는 데 사용한 코드는 다음 쪽에 있다.

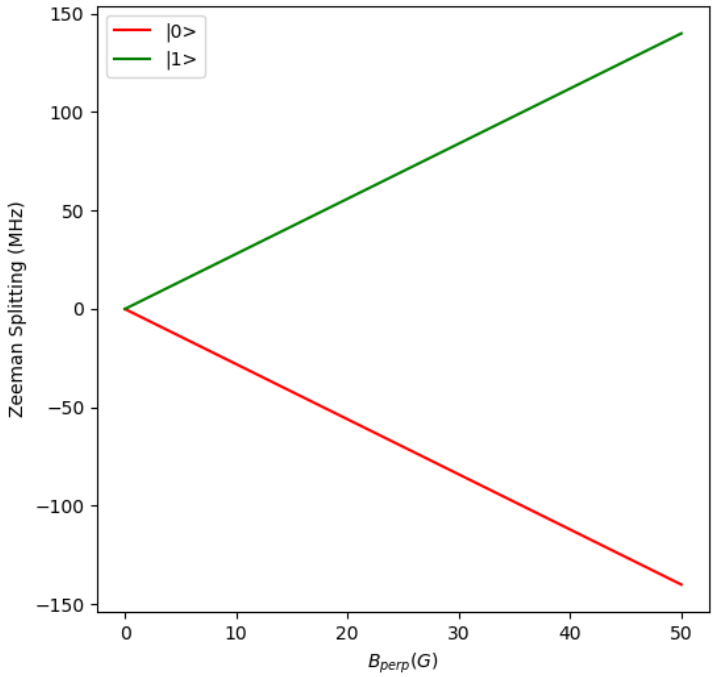

그림 3. Zeeman splitting.

```
from qutip import *
import numpy as np
import matplotlib.pyplot as plt

def compute(gamma, bperp):
    sx = sigmax()
    sz = sigmaz()
    H = gamma*bperp*sz
    evals, ekets = H.eigenstates()

    return evals
N = 10; end = 50 # G
gamma = 2.8 # MHz/G
bperp = np.linspace(0, end, N)
levels = np.zeros((2, N))

for i in range(N):
    evals = compute(gamma, bperp[i])
    levels[0,i] = evals[0]
    levels[1,i] = evals[1]


fig, ax = plt.subplots(figsize=(7, 7))
ax.plot(bperp, levels[0, :], color = 'red', label = '|0>')
ax.plot(bperp, levels[1, :], color = 'green', label = '|1>')
ax.set_xlabel('$B_{perp} (G)$')
ax.set_ylabel('Zeeman Splitting (MHz)')
ax.legend(loc='best')
plt.show()
```

2. Rabi oscillation

다음은 rabi oscillation이다. 여기서부턴 어떤 state의 **time evolution을 볼 것이므로 앞보단 복잡해진다.** 어떤 2 level system이 외부의 microwave의 영향을 받는 상황을 생각해보자.

$$\mathrm{H} = \frac{\omega_0}{2}\sigma_z + \Omega cos(\omega_m t + \phi)\sigma_x$$

여기서 ω_0는 2 level system의 energy gap이고, ω_m과 Ω, ϕ는 각각 microwave의 진동수와 세기(power), 그리고 위상(phase)이다. 이 해밀토니안은 time dependent하다. **여기에 $\boldsymbol{\omega_m}$으로 회전하는 rotating frame을 생각하고** rotating wave approximation (RWA)을 적용하면 time independent한 해밀토니안을 얻을 수 있다. RWA는 간단히 말해 rotating frame에서 빠르게 진동하는 항을 무시하는 것이다. 아래 계산에서는 $\omega_0 + \omega_m$의 진동수로 진동하는 항을 모두 무시했다(직접 계산해보면 어렵잖게 알 수 있다). 단, 이것은 $\Omega \ll \omega_0$가 만족될 때만 성립하는 근사임을 유의해야 한다.

$$H_{RWA} = \mathrm{U}\mathrm{H}U^{\dagger} + \mathrm{i}\dot{U}U^{\dagger} = \frac{\omega_0}{2}\sigma_z + \frac{\Omega}{2}\left(\cos\phi\sigma_x + sin\phi\sigma_y\right) - \frac{\omega_m}{2}\sigma_z$$
$$= \frac{\boldsymbol{\delta}}{\mathbf{2}}\boldsymbol{\sigma_z} + \frac{\boldsymbol{\Omega}}{\mathbf{2}}\left(\boldsymbol{\cos\phi\sigma_x} + \boldsymbol{sin\phi\sigma_y}\right)$$

$$\left(U = exp\left(\frac{\omega_m}{2}\sigma_z\right),\ \delta = \omega_0 - \omega_m = detune\right)$$

목표는 이 두 해밀토니안을 각각 sesolve()에 입력하여 time evolution을 계산하고 **시간에 따른 변화를 시각화**하는 것이다. QuTiP 공식 튜토리얼의 코드를 잘 이용하면 그렇게 어렵진 않다.

여기서부터는 필자가 작성한 코드로 만든 그림이 나오는데, 빨간색 화살표는 마지막 spin state를, 검은 점은 spin이 지나간 궤적을, 오른쪽 그래프들은 각각 σ_x, σ_y, σ_z의 기댓값을 나타낸다. 처음부터 이것과 같은 그림을 만들 필요는 없고 Bloch sphere 상에 화살표로 spin state를 표현할 수 있는 정도면 충분하다.

또한 필자의 경우 시간에 따른 spin state의 변화를 실시간으로 표시해주도록 했는데, 이를 하기 위해선 우선 Spyder의 설정을 조금 바꿔주어야 한다. Preference → IPython console → Graphics 에서 Activate support를 체크하면 된다.

이때 바로 밑의 Backend 설정이 Inline이면 안된다고 한다. 이렇게 하면 matplotlib으로 그림을 그릴 때 새 창을 띄워 실시간으로 업데이트 하게 되므로 시간에 따른 변화를 보기 좋고 Bolch sphere를 마우스로 클릭해 각도를 바꾸어가며 볼 수도 있다. 그림 4를 참조해서 적절히 설정했으면 IPython console을 새로 열어야 설정 변경이 반영된다. 무슨 말인지 모르겠다면 설정 후에 Spyder를 껐다 켜기만 해도 된다.

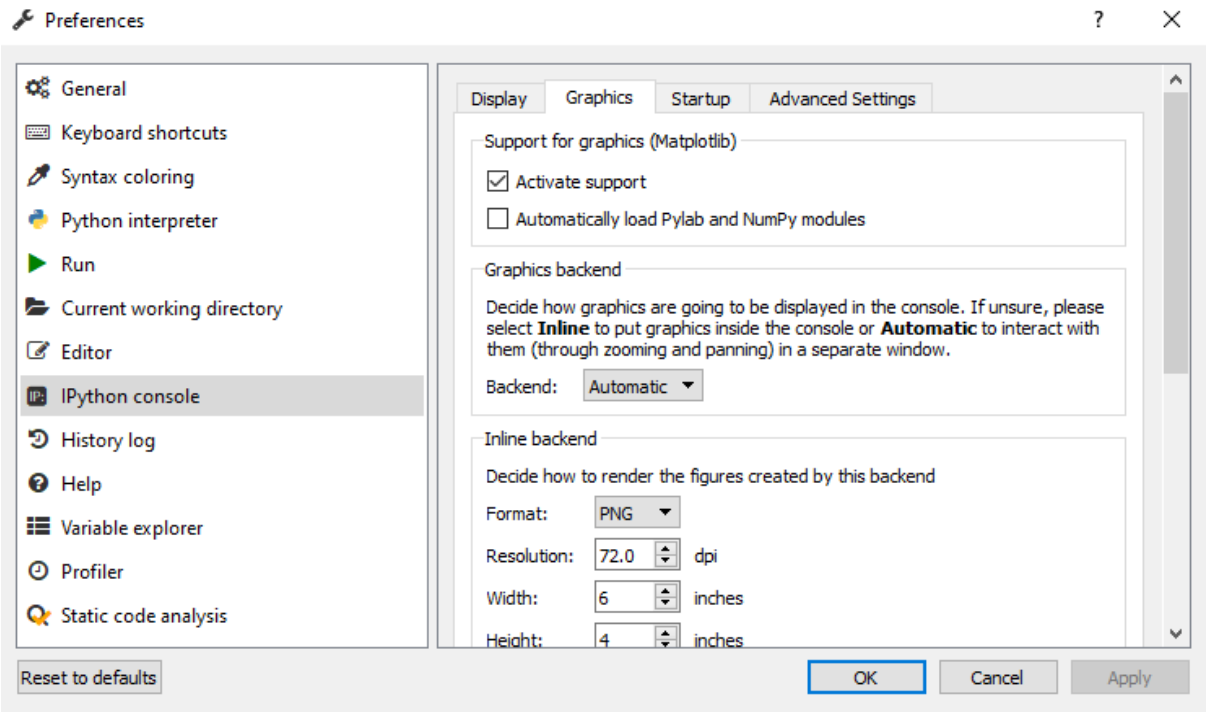

그림 4. Spyder 설정

우선 가장 간단한 것부터 시작해보자.

$$\mathrm{H} = \frac{\omega_0}{2}\sigma_z$$

이것은 외부의 microwave 등이 없이 2 level system의 energy gap만 존재할 때의 해밀토니안이다. $|0\rangle$에서 시작할 경우의 time evolution을 보면 그림 5와 같다. 필자는 $\omega_0 = 2\,GHz$를 사용했다.

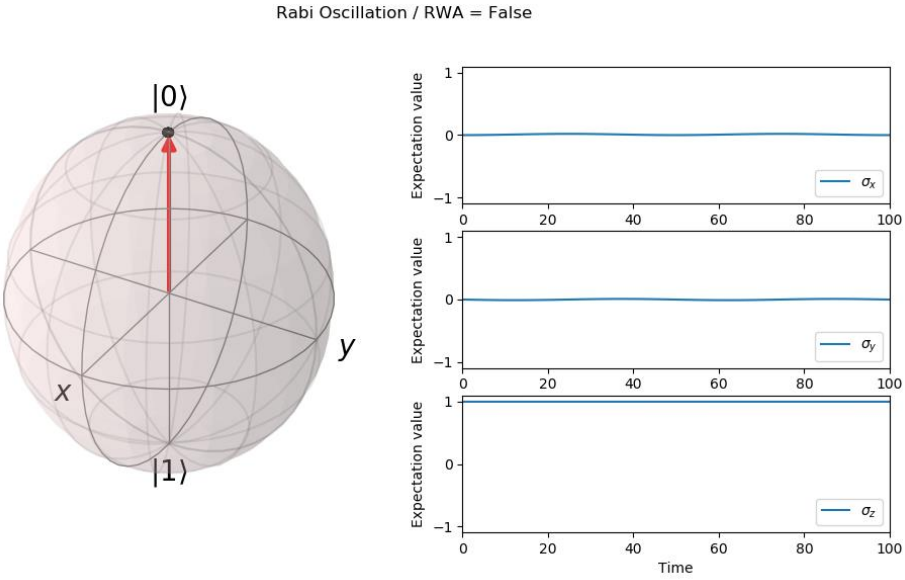

그림 5. Initial state = 0 state.

초기 상태에서 거의 변화가 없는 것으로 보인다. 그렇다면 이번엔 초기상태가 $|0\rangle$과 $|1\rangle$이 균등하게 중첩된 상태인 경우를 보자.

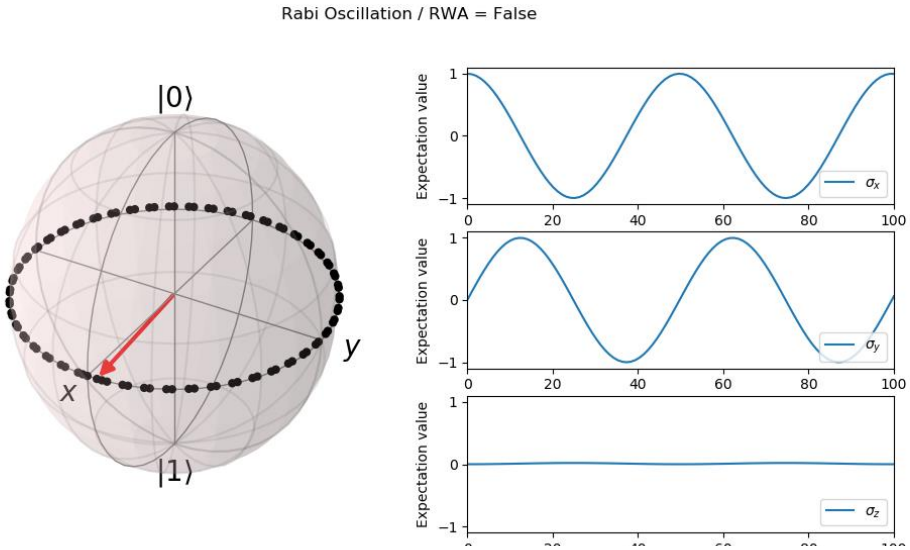

그림 6. Initial state = 1 state

이번엔 z축을 중심으로 회전한다는 것이 확실히 보인다. 이제 여기에 외부 microwave를 포함시켜보자.

$$\mathrm{H} = \frac{\omega_0}{2}\sigma_z + \Omega cos(\omega_m t + \phi)\sigma_x$$

$$\mathrm{H_{RWA}} = \frac{\delta}{2}\sigma_z + \frac{\Omega}{2}\left(\cos\phi\sigma_x + \sin\phi\sigma_y\right)$$

섹션 초입에서 제시한 해밀토니안이다. 두 해밀토니안을 각각 계산하면 그림 7,8과 같다. 시작은 0 state로, 마찬가지로 $\omega_0 = 2\,\mathrm{GHz}$를 사용했고 $\omega_m = \omega_0$이다.

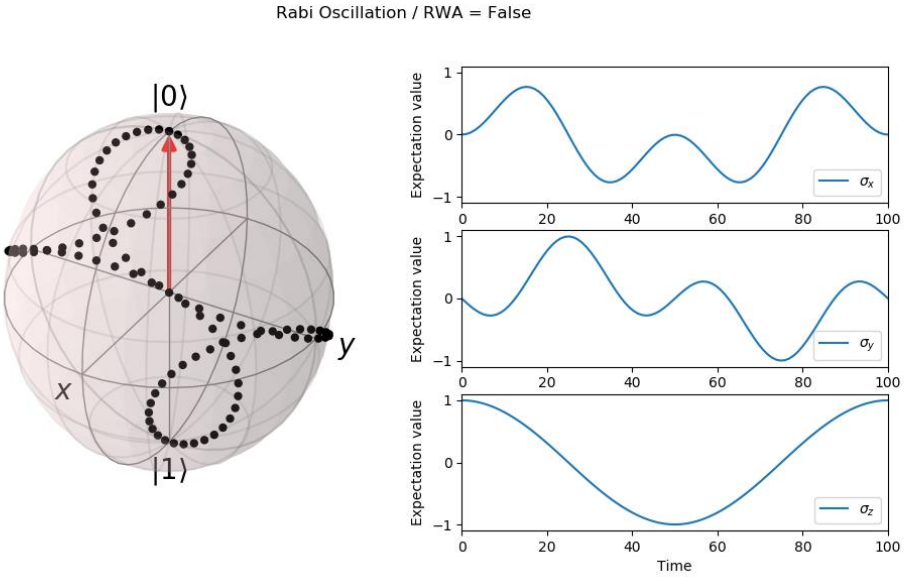

그림 8. Lab frame, without RWA.

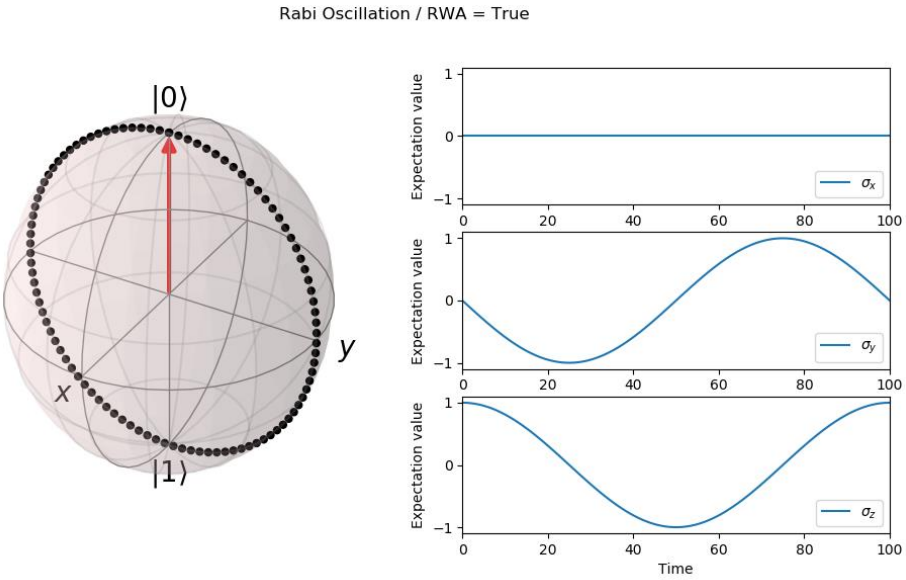

그림 7. Rotating frame, with RWA.

그림 7의 lab frame에서는 그림 4와 5에서 본 것과 같이 z축 중심의 회전이 섞여있고, 그림 8의 rotating frame에서는 이것이 보이지 않는다. 어쨌든 두 경우 모두 σ_z의 기댓값은 동일하게 변화하며 이는 spin state가 0과 1사이를 진동하고 있음을 알려준다. 이를 Rabi oscillation이라고 하며 이때의 진동수 $\sqrt{\delta^2 + \Omega^2}$를 Rabi frequency라고 한다.

여기까지 성공했다면 이제 ω_m을 값을 조금씩 바꾸어서 detune($\delta = \omega_0 - \omega_m$)을 바꾸어보자. 이 경우 detune이 커질수록 스핀이 0 state에서 잘 벗어나지 못하게 되는 것을 볼 수 있다. ϕ도 바꿔볼 수 있는데, detune에 비하면 변화가 크지 않다. 둘 모두 직접 해보고 변화를 관찰해보도록 하자.

다음 페이지엔 필자가 사용한 코드가 있다. 섹션 앞쪽에 설명한 Spyder 설정을 해두지 않으면 작동하지 않을 가능성이 있으므로 참고하자. 4개의 그림을 한 figure에 나타나도록 하려다보니 코드가 조금 복잡해져서 이해하기 힘들 수도 있다. 각주의 링크들[10][11]을 참조하면 약간의 도움이 될 것이다.

[10] https://stackoverflow.com/questions/50321057/how-to-show-qutip-bloch-sphere-as-onset-figure-on-the-bigger-plot-with-other-dat

[11] https://stackoverflow.com/questions/48744165/uneven-subplot-in-python

```
from qutip import *
import numpy as np
import matplotlib.pyplot as plt
import time

def TimeDepH(w0, wm, W, phi, psi0, tlist): # without RWA
    sx = sigmax(); sy = sigmay(); sz = sigmaz()
    H0 = w0 * sz / 2
    H1 = W * sx
    def H1_coeff(t, args):
        return np.cos(wm * t + phi)
    H = [H0,[H1,H1_coeff]]
    zero = Qobj([[1,0],[0,0]]) # probability of 0 state
    one = Qobj([[0,0],[0,1]]) # probabiliy of 1 state
    output = sesolve(H, psi0, tlist, [sx, sy, sz, zero, one])
    return output.expect[0], output.expect[1], output.expect[2], output.expect[3], output.expect[4]

def TimeIndepH(w0, wm, W, phi, psi0, tlist): # with RWA
    sx = sigmax(); sy = sigmay(); sz = sigmaz()
    H = ((w0-wm)/2)*sz + (W/2)*(np.cos(phi)*sx - np.sin(phi)*sy)
    zero = Qobj([[1,0],[0,0]])
    one = Qobj([[0,0],[0,1]])
    output = sesolve(H, psi0, tlist, [sx, sy, sz, zero, one])
    return output.expect[0], output.expect[1], output.expect[2], output.expect[3], output.expect[4]

w0 = 2*2*np.pi # energy gap, GHz
wm = 2*2*np.pi # external field freq.
W = 0.01*2*np.pi # external field powerS
```

```
phi = 0.5*np.pi # external field phase

N = 100 # number of steps
end = 100 # ns
tlist = np.linspace(0,end,N)

a = 1
psi0 = (a*basis(2,0)+(1-a)*basis(2,1))/np.sqrt(a**2+(1-a)**2) # initial state

RWA = False # turn RWA on and off
if(RWA):
    sx, sy, sz, zero, one = TimeIndepH(w0, wm, W, phi, psi0, tlist)
else:
    sx, sy, sz, zero, one = TimeDepH(w0, wm, W, phi, psi0, tlist)

fig = plt.figure(figsize=(12,6))
fig.suptitle(f'Rabi Oscillation / RWA = {RWA}') # a main title of the plots

ax = [] # make an empty list to store all the subplots
ax.append(fig.add_subplot(3,2,2)) # ax[0], sx
ax.append(fig.add_subplot(3,2,4)) # ax[1], sy
ax.append(fig.add_subplot(3,2,6)) # ax[3], sz
ax.append(fig.add_subplot(1,2,1, projection='3d')) # ax[3], Bloch sphere

b = Bloch(fig=fig, axes=ax[3]) # create Bloch sphere as one of the subplots
b.vector_color = ['r']
b.point_color = ['k']

for i in range(N):
```

```
for j in range(3): # clear the previous plot to draw a new one
    ax[j].clear()
b.clear()

ax[0].plot(tlist[:i+1], sx[:i+1], label='$\sigma_x$') # plot sx
ax[1].plot(tlist[:i+1], sy[:i+1], label='$\sigma_y$') # plot sy
ax[2].plot(tlist[:i+1], sz[:i+1], label='$\sigma_z$') # plot sz

b.add_vectors([sx[i], sy[i], sz[i]]) # display current spin state as a vector
b.add_points([sx[:i+1], sy[:i+1], sz[:i+1]]) # display trace of spin state
b.render(fig=fig, axes=ax[3]) # render the bloch sphere to the figure

for j in range(3):
    ax[j].set_xlabel('Time (ns)')
    ax[j].set_ylabel('Expectation value')
    ax[j].set_xlim([0,end])
    ax[j].set_ylim([-1.1,1.1])
    ax[j].legend(loc='lower right')

plt.draw()
plt.pause(0.001) # necessary to update the figure
```