technische universität
dortmund

SQuIDS: A Tool to Solve Time Evolution in finite dimensional (open) Quantum Systems

An Application to Neutrino Oscillations

arxiv:1412.3832

Dominik Hellmann

TU Dortmund
WG Päs

May 3, 2023

Motivation

Task: Solve time evolution of finite dimensional quantum (sub-)systems:

- Flavor oscillations
- Quantum computation
- Systems with finitely many energy levels
- Spins

Time evolution of closed quantum system: Schrödinger equation

$$i\frac{\partial}{\partial t}|\psi\rangle = \hat{H}|\psi\rangle \qquad (\hbar = 1) \tag{1}$$

Density matrices instead of state vectors

Often: finite dimensional system S coupled to a complicated (but uninteresting) environment E

$\rightarrow$ Get rid of Environment (keyword: partial trace)

$\rightarrow$ Just consider degrees of freedom of interest

Consequence: Decoherence

- Subsystem cannot be described by pure state $|\psi\rangle$
- Mixed state: Described by density matrix $\varrho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$

Example: Neutrino oscillations in matter

$S \simeq \mathbb{C}^3$	E
flavor degrees of freedom $\lvert\psi\rangle = \sum_{\alpha=e}^{\tau} \psi_\alpha \lvert\nu_\alpha\rangle$	all remaining d.o.f (momenta, spins, ...) $\rightarrow$ infinite dimensional

- We are not at all interested in E
- Only the ν flavor composition is interesting to us
- **But**: E significantly influences flavor d.o.f.

$\Rightarrow$ Need effective description!

Time evolution of the density matrix

Master equation(s): (multiple density matrices possible)

$$\frac{\mathrm{d}\varrho_j}{\mathrm{d}t} = -i[\hat{H}_j(t), \varrho_j(t)] + \{\Gamma_j(t), \varrho_j(t)\} + F_j[\{\varrho_k\}_k, t]$$

- Why multiple ϱ_j? E.g.: One per energy bin!
- $\hat{H} = \hat{H}_0 + \hat{H}_1(t)$: Unitary evolution
- Γ: Decoherence
- F: Other non-linear effects (coupling between ϱ_j)

Simple Example: ν Oscillations in Vacuum

Neutrino Experiment:

- Fixed baseline L
- N energy bins $\{E_j\}_j$
- $\hat{H}^j = \hat{H}_0^j = E_j \cdot \mathbb{I} + \frac{1}{2E_j}\mathbb{M}^2$
- $\Gamma \equiv 0$
- $F \equiv 0$
- $\varrho_j(t) = \sum_{\alpha=e}^{\tau} \phi_\alpha^j(t) |\nu_\alpha\rangle\langle\nu_\alpha|$

$$\mathbb{M} = \sum_{j,k=1}^{3} (\mathbb{M}_0)_{jk} |\nu_j\rangle\langle\nu_k|$$

$$= \sum_{\alpha,\beta=e}^{\tau} (\mathbb{M}_1)_{\alpha\beta} |\nu_\alpha\rangle\langle\nu_\beta|$$

$$\mathbb{M}_0 = \begin{pmatrix} m_1 & 0 & 0 \\ 0 & m_2 & 0 \\ 0 & 0 & m_3 \end{pmatrix}$$

$$\mathbb{M}_1 = U_{\mathrm{PMNS}}^\dagger \mathbb{M}_1 U_{\mathrm{PMNS}}$$

Some Important Considerations

The master equation simplifies to

$$\frac{\mathrm{d}\varrho_j}{\mathrm{d}t} = -i[\hat{H}_0^j, \varrho_j(t)]$$

Further simplifications

- Consider in mass basis: $\hat{H}_0^j$ is diagonal
- Depends only on commutator!
  - $[A, \mathbb{I}] = 0 \Rightarrow [\hat{H}_0^j, \varrho_j(t)] = [\hat{H}_0^j - \epsilon_j \mathbb{I}, \varrho_j(t)]$
  - $\epsilon_j := E_j + m_1^2/2E_j$

$$\tilde{H}^j = \frac{1}{2E_j} \begin{pmatrix} 0 & 0 & 0 \\ 0 & \Delta m_{21}^2 & 0 \\ 0 & 0 & \Delta m_{31}^2 \end{pmatrix}$$

Some Important Considerations (ctd.)

Summary

What did we learn so far (in general)

1. We passed to density matrix formulation (allows for mixed states)
2. Formulated master equation
3. Can subtract a part prop. to identity from $\hat{H}$ (only energy diff. important)
4. Can solve $\hat{H}_0$ exactly (interaction picture) $\varrho \to e^{-i\hat{H}_0\Delta T}\varrho e^{i\hat{H}_0\Delta T}$

Overview

Const

SU Vector

SQuIDS

Clone the Repo!

Git Repository includes all needed files (slides, code templates)

Instructions

cd to the location where you want to place the repo

```
git clone https://github.com/B0bsen/sm_to_bsm_neutrino.git
cd sm_to_bsm_neutrino
```

Const class exercise

1. Declare a default constructed const class object
2. Answer the following questions:
   - 2.1 How many eV^{-1} correspond to $300\,\mathrm{km}$
   - 2.2 How many radians correspond to $25°$
   - 2.3 If you are 24 years old, how many eV^{-1} are you old?
3. Set the mixing parameters for three neutrino generations to:
   - $\theta_{12} = 33.48°$
   - $\theta_{13} = 8.55°$
   - $\theta_{23} = 42.3°$
4. Set the energy differences to:
   - $\Delta m^2_{21} = 7.5 \cdot 10^{-5}\,\mathrm{eV}^2$
   - $\Delta m^2_{31} = 2.45 \cdot 10^{-3}\,\mathrm{eV}^2$

SU vector exercise

1. Declare an empty SU vector corresponding to a 3D Hilbert space
2. Initialize an array of projectors for the three mass eigenstates (B_0)
3. Rotate them to the flavor basis (B_1)
4. Initialize a SU vector corresponding to the matrix (B_0)

$$\Delta\mathbb{M}^2 := \begin{pmatrix} 0 & 0 & 0 \\ 0 & \Delta m^2_{21} & 0 \\ 0 & 0 & m^2_{31} \end{pmatrix} \tag{2}$$

SQuIDS application: Neutrino oscillations in vacuum

Exercises

Exercises

BACK UP

Installation

What do we need for this tutorial?

- A unix-like (sub-)system
  - Linux
  - Mac (+ Xcode developer tools!)
  - On Windows: WSL
- A C++ compiler
- Make, wget, Git

Use scripts `install_gsl.sh` and `install_SQuIDS.sh` from the repo!

Installation (GSL)

```
cd $HOME
mkdir -p smToBsmLibs/gsl
wget ftp://ftp.gnu.org/gnu/gsl/gsl-latest.tar.gz
tar -zxvf gsl-latest.tar.gz
rm gsl-latest.tar.gz
cd $(find gsl-* | head -n 1)
./configure --prefix=$HOME/smToBsmLibs/gsl
make
make check
make install
LD_LIBRARY_PATH=$HOME/smToBsmLibs/gsl/lib:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH
cd $HOME
rm -rf $(find gsl-* | head -n 1)
```

Installation (SQuIDS)

```
cd $HOME
mkdir -p smToBsmLibs/SQuIDS
git clone https://github.com/jsalvado/SQuIDS.git
cd $(find SQuIDS* | head -n 1)
./configure --with-gsl-incdir=$HOME/smToBsmLibs/gsl/include \
--with-gsl-libdir=$HOME/smToBsmLibs/gsl/lib \
--prefix=$HOME/smToBsmLibs/SQuIDS
make
make test
make install
LD_LIBRARY_PATH=$HOME/smToBsmLibs/SQuIDS/lib:$LD_LIBRARY_PATH # linux only
export LD_LIBRARY_PATH # linux only
cd $HOME
rm -rf $(find SQuIDS* | head -n 1)
```