Universität Augsburg
Mathematisch-Naturwissenschaftlich-Technische Fakultät

# Computer aided Analytical Calculations for Physical Many-Body Problems

## - Project Work Presentation -


**Jonas Kell**
Chair for theoretical Physics III


$15^{\text{th}}$ of Mai 2024

2025-02-11

Computer aided calculations

1. Welcome
2. Presentation of "Project Work" (Projektarbeit)
3. Runs parallel to the "Practical Training" (Fachpraktikum)
4. Presentation in this Group-Slot

# Outline







Outline
1 Introduction of the problem

1. Introduction of theme worked on in Project-Work, Practical Training and Master Thesis
   - The mathematical Problem of Many-Body Physics
2. Focus of the report is on the mathematical details. I have a history in Computer-Science, therefore would like here to present some techniques I learned in my work as well as studies as a Computer scientist, that I think might be quite useful to apply in the context of such a working group.
   - The computational/notational problem of "doing maths easily"
3. What was there to make it easier (Off the shelf solutions)
4. What I did:
   - Math Manipulator
   - Tricks to use to improve your Python
   - Latex for presentations
5. Gist: after I (somewhat of a computer scientist by trade) learned the process of being a theoretical physicist, I want to bring back some tools/workflows to maybe improve someones life here at TP III

# Outline







Outline
1 Introduction of the problem
2 Solutions: Off the shelf

1. Introduction of theme worked on in Project-Work, Practical Training and Master Thesis
   - The mathematical Problem of Many-Body Physics
2. Focus of the report is on the mathematical details. I have a history in Computer-Science, therefore would like here to present some techniques I learned in my work as well as studies as a Computer scientist, that I think might be quite useful to apply in the context of such a working group.
   - The computational/notational problem of "doing maths easily"
3. What was there to make it easier (Off the shelf solutions)
4. What I did:
   - Math Manipulator
   - Tricks to use to improve your Python
   - Latex for presentations
5. Gist: after I (somewhat of a computer scientist by trade) learned the process of being a theoretical physicist, I want to bring back some tools/workflows to maybe improve someones life here at TP III

# Outline







Outline

1 Introduction of the problem
2 Solutions: Off the shelf
3 Math-Manipulator
- What did I do?
- How can you use it
- Interactive Demonstration

1. Introduction of theme worked on in Project-Work, Practical Training and Master Thesis
   - The mathematical Problem of Many-Body Physics
2. Focus of the report is on the mathematical details. I have a history in Computer-Science, therefore would like here to present some techniques I learned in my work as well as studies as a Computer scientist, that I think might be quite useful to apply in the context of such a working group.
   - The computational/notational problem of "doing maths easily"
3. What was there to make it easier (Off the shelf solutions)
4. What I did:
   - Math Manipulator
   - Tricks to use to improve your Python
   - Latex for presentations
5. Gist: after I (somewhat of a computer scientist by trade) learned the process of being a theoretical physicist, I want to bring back some tools/workflows to maybe improve someones life here at TP III

# Outline







Outline
1 Introduction of the problem
2 Solutions: Off the shelf
3 Math-Manipulator
What did I do?
How can you use it
Interactive Demonstration
4 Custom Python Scripts (SymPy)

1. Introduction of theme worked on in Project-Work, Practical Training and Master Thesis
   – The mathematical Problem of Many-Body Physics
2. Focus of the report is on the mathematical details. I have a history in Computer-Science, therefore would like here to present some techniques I learned in my work as well as studies as a Computer scientist, that I think might be quite useful to apply in the context of such a working group.
   – The computational/notational problem of "doing maths easily"
3. What was there to make it easier (Off the shelf solutions)
4. What I did:
   – Math Manipulator
   – Tricks to use to improve your Python
   – Latex for presentations
5. Gist: after I (somewhat of a computer scientist by trade) learned the process of being a theoretical physicist, I want to bring back some tools/workflows to maybe improve someones life here at TP III

# Outline







Outline

1 Introduction of the problem
2 Solutions: Off the shelf
3 Math-Manipulator
- What did I do?
- How can you use it
- Interactive Demonstration
4 Custom Python Scripts (SymPy)
5 Presentations? - Custom Beamer Template

1. Introduction of theme worked on in Project-Work, Practical Training and Master Thesis
   - The mathematical Problem of Many-Body Physics
2. Focus of the report is on the mathematical details. I have a history in Computer-Science, therefore would like here to present some techniques I learned in my work as well as studies as a Computer scientist, that I think might be quite useful to apply in the context of such a working group.
   - The computational/notational problem of "doing maths easily"
3. What was there to make it easier (Off the shelf solutions)
4. What I did:
   - Math Manipulator
   - Tricks to use to improve your Python
   - Latex for presentations
5. Gist: after I (somewhat of a computer scientist by trade) learned the process of being a theoretical physicist, I want to bring back some tools/workflows to maybe improve someones life here at TP III

# Physics of what we calculate

## Introduction of the problem

- Content of my Practical Training (Fachpraktikum)

$$\mathcal{H} = \mathcal{H}_0 + \hat{\mathrm{V}}$$

$$\mathcal{H}_0 = U \cdot \sum_l \hat{\mathrm{n}}_{l,\uparrow}\hat{\mathrm{n}}_{l,\downarrow} + \sum_{l,\sigma} \underbrace{\left(\vec{E}\cdot\vec{r}_l\right)}_{\varepsilon_l} \hat{\mathrm{n}}_{l,\sigma}$$

$$\hat{\mathrm{V}} = -J \cdot \sum_{\langle l,m\rangle,\sigma} \left(\hat{\mathrm{c}}^\dagger_{l,\sigma}\hat{\mathrm{c}}_{m,\sigma} + \hat{\mathrm{c}}^\dagger_{m,\sigma}\hat{\mathrm{c}}_{l,\sigma}\right)$$





Physics of what we calculate
Introduction of the problem
Content of my Practical Training (Fachpraktikum)

1. Hubbard-Model Hamiltonian with exterally applied electric field
2. Hard-Core Bosinic Operators (will be explained in later slide, do not need to elaborate on)
3. Spin-dependent lattice Model
4. External electric field acts symmetric on both spin directions
   - This will be important later, multiple times
   - All terms always need to be symmetric in terms of spin up/down

# Physics of what we calculate

## Introduction of the problem

- Content of my Practical Training (Fachpraktikum)
- Hubbard Model Hamiltonian

$$\mathcal{H} = \mathcal{H}_0 + \hat{\mathrm{V}}$$

$$\mathcal{H}_0 = U \cdot \sum_{l} \hat{\mathrm{n}}_{l,\uparrow} \hat{\mathrm{n}}_{l,\downarrow} + \sum_{l,\sigma} \underbrace{\left(\vec{E} \cdot \vec{r}_l\right)}_{\varepsilon_l} \hat{\mathrm{n}}_{l,\sigma}$$

$$\hat{\mathrm{V}} = -J \cdot \sum_{\langle l,m \rangle,\sigma} \left(\hat{\mathrm{c}}^{\dagger}_{l,\sigma} \hat{\mathrm{c}}_{m,\sigma} + \hat{\mathrm{c}}^{\dagger}_{m,\sigma} \hat{\mathrm{c}}_{l,\sigma}\right)$$





Physics of what we calculate
Introduction of the problem
Content of my Practical Training (Fachpraktikum)
Hubbard Model Hamiltonian

1. Hubbard-Model Hamiltonian with exterally applied electric field
2. Hard-Core Bosinic Operators (will be explained in later slide, do not need to elaborate on)
3. Spin-dependent lattice Model
4. External electric field acts symmetric on both spin directions
   - This will be important later, multiple times
   - All terms always need to be symmetric in terms of spin up/down

## Physics of what we calculate

### Introduction of the problem

- Content of my Practical Training (Fachpraktikum)
- Hubbard Model Hamiltonian
- System under influence of external electric field

$$\mathcal{H} = \mathcal{H}_0 + \hat{\mathrm{V}}$$

$$\mathcal{H}_0 = U \cdot \sum_{l} \hat{\mathrm{n}}_{l,\uparrow}\hat{\mathrm{n}}_{l,\downarrow} + \sum_{l,\sigma} \underbrace{\left(\vec{E}\cdot\vec{r}_l\right)}_{\varepsilon_l} \hat{\mathrm{n}}_{l,\sigma}$$

$$\hat{\mathrm{V}} = -J \cdot \sum_{\langle l,m\rangle,\sigma} \left(\hat{\mathrm{c}}^{\dagger}_{l,\sigma}\hat{\mathrm{c}}_{m,\sigma} + \hat{\mathrm{c}}^{\dagger}_{m,\sigma}\hat{\mathrm{c}}_{l,\sigma}\right)$$



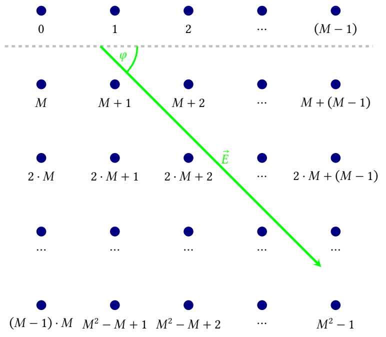


1. Hubbard-Model Hamiltonian with exterally applied electric field
2. Hard-Core Bosinic Operators (will be explained in later slide, do not need to elaborate on)
3. Spin-dependent lattice Model
4. External electric field acts symmetric on both spin directions
   - This will be important later, multiple times
   - All terms always need to be symmetric in terms of spin up/down

# Physics of what we calculate

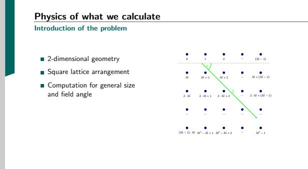
## Introduction of the problem

- 2-dimensional geometry
- Square lattice arrangement
- Computation for general size and field angle







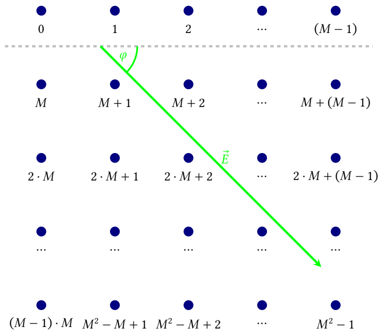
1. 2-dimensional square geometry
2. Program supports already for general size inputs
3. Thanks to the Monte-Carlo Sampling we can even efficiently evaluate larger systems already

# Physics of what we calculate

## Introduction of the problem

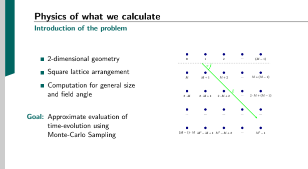
- 2-dimensional geometry
- Square lattice arrangement
- Computation for general size and field angle

**Goal:** Approximate evaluation of time-evolution using Monte-Carlo Sampling







1. 2-dimensional square geometry
2. Program supports already for general size inputs
3. Thanks to the Monte-Carlo Sampling we can even efficiently evaluate larger systems already

# Physics of what we calculate

## Introduction of the problem

- Hard-Core Bosonic operators

$$\left[\hat{c}_{l,\sigma}, \hat{c}_{l',\sigma'}\right] = \left[\hat{c}^{\dagger}_{l,\sigma}, \hat{c}^{\dagger}_{l',\sigma'}\right] = 0$$
$$\left[\hat{c}_{l,\sigma}, \hat{c}^{\dagger}_{l',\sigma'}\right] = \delta(l,l') \cdot \delta(\sigma,\sigma')$$





Physics of what we calculate
Introduction of the problem
Hard-Core Bosonic operators

1. Operators are hard-core bosonic
   - standard commutation relations
   - occupations are only either zero ore one
2. fermionic operators (anti-commutation) would work equivalent
3. other notation to rewrite the two spin degrees of freedom into two operators of different dof makes it more readable and comfortable to apply computationally in some cases

# Physics of what we calculate

## Introduction of the problem

$$\left[\hat{c}_{l,\sigma}, \hat{c}_{l',\sigma'}\right] = \left[\hat{c}^\dagger_{l,\sigma}, \hat{c}^\dagger_{l',\sigma'}\right] = 0$$
$$\left[\hat{c}_{l,\sigma}, \hat{c}^\dagger_{l',\sigma'}\right] = \delta(l,l') \cdot \delta(\sigma,\sigma')$$

- Hard-Core Bosonic operators
  - (Would work analogously with Fermionic operators)





Physics of what we calculate
Introduction of the problem
$[\hat{c}_{l,\sigma}, \hat{c}_{l',\sigma'}] = [\hat{c}^\dagger_{l,\sigma}, \hat{c}^\dagger_{l',\sigma'}] = 0$
$[\hat{c}_{l,\sigma}, \hat{c}^\dagger_{l',\sigma'}] = \delta(l,l') \cdot \delta(\sigma,\sigma')$
Hard-Core Bosonic operators
(Would work analogously with Fermionic operators)

1. Operators are hard-core bosonic
   - standard commutation relations
   - occupations are only either zero ore one
2. fermionic operators (anti-commutation) would work equivalent
3. other notation to rewrite the two spin degrees of freedom into two operators of different dof makes it more readable and comfortable to apply computationally in some cases

# Physics of what we calculate

## Introduction of the problem

- Hard-Core Bosonic operators
  - (Would work analogously with Fermionic operators)
- Two spin-degrees of freedom rewritten in alternate notation

$$\left[\hat{c}_{l,\sigma}, \hat{c}_{l',\sigma'}\right] = \left[\hat{c}^\dagger_{l,\sigma}, \hat{c}^\dagger_{l',\sigma'}\right] = 0$$

$$\left[\hat{c}_{l,\sigma}, \hat{c}^\dagger_{l',\sigma'}\right] = \delta(l,l') \cdot \delta(\sigma,\sigma')$$

$$\hat{c}^{(\dagger)}_{l,\uparrow} \leftrightarrow \hat{c}^{(\dagger)}_{l} \qquad \hat{c}^{(\dagger)}_{l,\downarrow} \leftrightarrow \hat{d}^{(\dagger)}_{l}$$

$$\mathcal{H}_0 = U \cdot \sum_l \hat{c}^\dagger_l \hat{c}_l \hat{d}^\dagger_l \hat{d}_l + \sum_l \varepsilon_l \hat{c}^\dagger_l \hat{c}_l \sum_l \varepsilon_l \hat{d}^\dagger_l \hat{d}_l$$

$$\hat{V} = -J \cdot \sum_{\langle l,m \rangle} \left( \hat{c}^\dagger_l \hat{c}_m + \hat{c}^\dagger_m \hat{c}_l + \hat{d}^\dagger_l \hat{d}_m + \hat{d}^\dagger_m \hat{d}_l \right)$$





1. Operators are hard-core bosonic
   - standard commutation relations
   - occupations are only either zero ore one
2. fermionic operators (anti-commutation) would work equivalent
3. other notation to rewrite the two spin degrees of freedom into two operators of different dof makes it more readable and comfortable to apply computationally in some cases

# Working on paper: A Computer-Scientists view

## Introduction of the problem

Doing Theoretical Physics often requires lengthy analytical calculations





Working on paper: A Computer-Scientists view
Introduction of the problem
Doing Theoretical Physics often requires lengthy analytical calculations

1. Comparison of advantages/disatvantages of doing calculations "by hand"
2. Advantages
   – No skills required to learn, writing is natural, every new symbol can just be written
   – Operational Libery (cross out stuff as please, swap whatever without "proof")
   – For completely different approaches no new tool is needed (lern that integral cannot be solved with a specific strategy that you tool doesn't support: out of luck)
   – Hard to loose access to taken notes, produce them without internet/power
3. Disatvantages
   – Many tasks are repetetive, trivial and unfunny to do, paper cannot automize
   – Sign errores, copy errors happen too often
   – Sharing and archiving is non-standart. Many standarts exist for e.g. sharing latex
   – No version-control (git): hard for people that are used to it like me. Cannot just do stuff and later revert if it didn't work
   – Need to write the pretty version in the computer anyway

# Working on paper: A Computer-Scientists view

## Introduction of the problem

Doing Theoretical Physics often requires lengthy analytical calculations

**Advantages** of working on paper:

- No barrier to entry (required Skill & technology)
- Maximum liberty how to operate
- Fast iteration for high variation workload
- "Offline" available





Working on paper: A Computer-Scientists view
Introduction of the problem
Doing Theoretical Physics often requires lengthy analytical calculations
Advantages of working on paper:
No barrier to entry (required Skill & technology)
Maximum liberty how to operate
Fast iteration for high variation workload
"Offline" available

1. Comparison of advantages/disatvantages of doing calculations "by hand"
2. Advantages
   - No skills required to learn, writing is natural, every new symbol can just be written
   - Operational Libery (cross out stuff as please, swap whatever without "proof")
   - For completely different approaches no new tool is needed (lern that integral cannot be solved with a specific strategy that you tool doesn't support: out of luck)
   - Hard to loose access to taken notes, produce them without internet/power
3. Disatvantages
   - Many tasks are repetetive, trivial and unfunny to do, paper cannot automize
   - Sign errores, copy errors happen too often
   - Sharing and archiving is non-standart. Many standarts exist for e.g. sharing latex
   - No version-control (git): hard for people that are used to it like me. Cannot just do stuff and later revert if it didn't work
   - Need to write the pretty version in the computer anyway

# Working on paper: A Computer-Scientists view

## Introduction of the problem

Doing Theoretical Physics often requires lengthy analytical calculations

**Advantages** of working on paper:

- No barrier to entry (required Skill & technology)
- Maximum liberty how to operate
- Fast iteration for high variation workload
- "Offline" available

**Problems** of working on paper:

- Repetitive tasks not automizable
- Error-prone and time-consuming to fix mistakes
- Difficult & non-standard to share/archive
- No version-control
- Need to produce final version anyway





Working on paper: A Computer-Scientists view
Introduction of the problem
Doing Theoretical Physics often requires lengthy analytical calculations
Advantages of working on paper: No barrier to entry (required Skill & technology); Maximum liberty how to operate; Fast iteration for high variation workload; "Offline" available
Problems of working on paper: Repetitive tasks not automizable; Error-prone and time-consuming to fix mistakes; Difficult & non-standard to share/archive; No version-control; Need to produce final version anyway

1. Comparison of advantages/disatvantages of doing calculations "by hand"
2. Advantages
   - No skills required to learn, writing is natural, every new symbol can just be written
   - Operational Libery (cross out stuff as please, swap whatever without "proof")
   - For completely different approaches no new tool is needed (lern that integral cannot be solved with a specific strategy that you tool doesn't support: out of luck)
   - Hard to loose access to taken notes, produce them without internet/power
3. Disatvantages
   - Many tasks are repetetive, trivial and unfunny to do, paper cannot automize
   - Sign errores, copy errors happen too often
   - Sharing and archiving is non-standart. Many standarts exist for e.g. sharing latex
   - No version-control (git): hard for people that are used to it like me. Cannot just do stuff and later revert if it didn't work
   - Need to write the pretty version in the computer anyway

# Examples from "Theoretical Solid State Physics"

## Introduction of the problem

$$[S,H] = [c_1^\dagger c_2 - c_2^\dagger c_1,\ \varepsilon c_1^\dagger c_1 + \Delta(c_1^\dagger c_2 + c_2^\dagger c_1)] \qquad \nearrow [c_1^\dagger c_2, c_1^\dagger c_2] = 0, \text{ obvious, usw.}$$

$$= \varepsilon [c_1^\dagger c_2, c_1^\dagger c_1] - \varepsilon [c_2^\dagger c_1 - c_1^\dagger c_1] + \Delta [c_1^\dagger c_2, c_2^\dagger c_1] - \Delta [c_2^\dagger c_1, c_1^\dagger c_2]$$

$$c_1^\dagger \underbrace{c_2 c_1^\dagger}_{=-c_1^\dagger c_2} c_1 - c_1^\dagger \underbrace{c_1 c_1^\dagger}_{=1-c_1^\dagger c_1} c_2 = c_1^\dagger c_2 \quad \Big| \quad c_2^\dagger c_1 \quad \Big| \quad c_1^\dagger \underbrace{c_2 c_2^\dagger}_{1-c_2^\dagger c_2} c_1 - c_2^\dagger \underbrace{c_1 c_1^\dagger}_{1-c_1^\dagger c_1} c_2 =$$

$$= c_1^\dagger c_1 - c_1^\dagger c_2^\dagger \underbrace{c_2 c_1}_{-c_1 c_2} - c_2^\dagger c_2 + \underbrace{c_2^\dagger c_1^\dagger}_{-c_1^\dagger c_2^\dagger} c_1 c_2 =$$

$$= c_1^\dagger c_1 + c_1^\dagger c_2^\dagger c_1 c_2 - c_2^\dagger c_2 - c_1^\dagger c_2^\dagger c_1 c_2 = c_1^\dagger c_1 - c_2^\dagger c_2$$

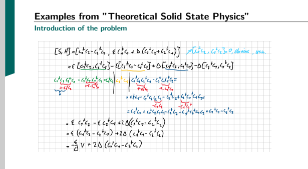
$$= \varepsilon c_1^\dagger c_2 - \varepsilon c_2^\dagger c_1 + 2\Delta (c_1^\dagger c_1 - c_2^\dagger c_2)$$

$$= \varepsilon (c_1^\dagger c_2 - c_2^\dagger c_1) + 2\Delta (c_1^\dagger c_1 - c_2^\dagger c_2)$$

$$= -\frac{\varepsilon}{\Delta} V + 2\Delta (c_1^\dagger c_1 - c_2^\dagger c_2)$$







1. From the lecture "Theoretical Solid State Physics" (Theoretische Festkörperphysik), helt by Markus Heyl in WS 22/23
2. Sheet 3, Problem 2: Schrieffer-Wolf transformation
3. Rather simple calculation, still contains sing mistakes (that here luckily averaged out)

# Examples from "Theoretical Solid State Physics"

## Introduction of the problem

Find transform

$$c_{k\sigma}^\dagger(t) = e^{iHt} c_{k\sigma}^\dagger e^{-iHt} \rightarrow \text{complicated}$$

$$\tilde{c}_{k\sigma}^\dagger(t) = e^{iHt} \tilde{c}_{k\sigma}^\dagger e^{-iHt} \rightarrow \text{easier}$$

$$\tilde{c}_{k\sigma}^\dagger = e^S c_{k\sigma}^\dagger e^{-S} \approx c_{k\sigma}^\dagger + [S, c_{k\sigma}^\dagger] + O(U^2)$$

$$[S, c_{k\sigma}^\dagger] = \frac{U}{N} \sum_{k_1 \dots k_4}{}' \frac{\delta(k_1 - k_2 + k_3 - k_4)}{\varepsilon_{k_1} - \varepsilon_{k_2} + \varepsilon_{k_3} - \varepsilon_{k_4}} \underbrace{[c_{k_1\uparrow}^\dagger c_{k_2\uparrow} c_{k_3\downarrow}^\dagger c_{k_4\downarrow}, c_{k\sigma}]}_{c_{k_1\uparrow}^\dagger c_{k_3\downarrow}^\dagger c_{k_4\downarrow} \delta_{kk_2} \delta_{\sigma\uparrow} + c_{k_1\uparrow}^\dagger c_{k_2\uparrow} c_{k_3\downarrow}^\dagger \delta_{kk_4} \delta_{\sigma\downarrow}} =$$

$$\tilde{c}_{k\sigma}^\dagger = e^S c_{k\sigma}^\dagger e^{-S} = c_{k\sigma}^\dagger + \frac{U}{N} \sum_{k_1 \dots k_4}{}' \frac{\delta(k_1 - k_2 + k_3 - k_4)}{\varepsilon_{k_1} - \varepsilon_{k_2} + \varepsilon_{k_3} - \varepsilon_{k_4}}$$

change index $k_2 \leftrightarrow k_4$, $k_1 \leftrightarrow k_3$

$$= c_{k\sigma}^\dagger + \frac{U}{N} \sum_{k_1 \dots k_4}{}' \frac{\delta(k_1 - k_2 + k_3 - k_4)}{\varepsilon_{k_1} - \varepsilon_{k_2} + \varepsilon_{k_3} - \varepsilon_{k_4}} \delta_{kk_2} c_{k_1\sigma}^\dagger c_{k_3\bar{\sigma}}^\dagger c_{k_4\bar{\sigma}}$$

$k_4 \rightarrow k_1 - k_2 + k_3$

$$= c_{k\sigma}^\dagger + \frac{U}{N} \sum_{k_1 \dots k_4}{}' \frac{1}{\varepsilon_{k_1} - \varepsilon_{k} + \varepsilon_{k_3} - \varepsilon_{k_1 - k + k_3}} c_{k\sigma}^\dagger c_{k_3\bar{\sigma}}^\dagger c_{k_1 - k + k_3\,\bar{\sigma}}$$

ii)

$$\bar{H} = uHu^\dagger = e^S H e^{-S} = H + [S, H_0] + O(U^2)$$

$[S, V] \rightarrow$ second order

$$[S, H_0] = \frac{U}{N} \sum_{k_1 \dots k_4}{}' \frac{\delta(k_1 - k_2 + k_3 - k_4)}{\varepsilon_{k_1} - \varepsilon_{k_2} + \varepsilon_{k_3} - \varepsilon_{k_4}} \underbrace{[c_{k_1\uparrow}^\dagger c_{k_2\uparrow} c_{k_3\downarrow}^\dagger c_{k_4\downarrow}, \sum_{k\sigma} \varepsilon_k n_{k\sigma}]}_{-(\varepsilon_{k_1} - \varepsilon_{k_2} + \varepsilon_{k_3} - \varepsilon_{k_4}) c_{k_1\uparrow}^\dagger c_{k_2\uparrow} c_{k_3\downarrow}^\dagger c_{k_4\downarrow}}$$

$\rightarrow$ use $[c_{k\sigma}^\dagger, n_{k'\sigma'}]$

$$u = e^S, \quad S = \frac{1}{N} \sum_{k_1 \dots k_4}{}' \delta(k_1 - k_2 + k_3 - k_4) \frac{U}{\epsilon_{k_1} - \epsilon_{k_2} + \epsilon_{k_3} - \epsilon_{k_4}} c_{k_1\uparrow}^\dagger c_{k_2\uparrow} c_{k_3\downarrow}^\dagger c_{k_4\downarrow} \quad (2)$$

we can eliminate to leading order in $U$ the interaction term. Notice that the summation $\sum'_{k_1 \dots k_4}$ only involves those contributions for which $\epsilon_{k_1} - \epsilon_{k_2} + \epsilon_{k_3} - \epsilon_{k_4} \neq 0$.

a) Calculate to the leading order in $U$ the time evolution $c_{k\sigma}^\dagger(t)$ of single fermion operators in Heisenberg picture:

$$c_{k\sigma}^\dagger(t) = e^{iHt} c_{k\sigma}^\dagger e^{-iHt} \quad (3)$$

i) First transform both the Hamiltonian $H$ and $c_{k\sigma}^\dagger$ with $u$:

$$\bar{H} = uHu^\dagger, \quad \tilde{c}_{k\sigma}^\dagger = u c_{k\sigma}^\dagger u^\dagger \quad (4)$$

(see previous exercise sheet)

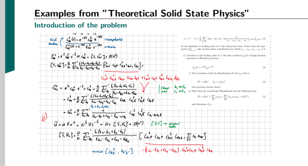
ii) Show that the transformed Hamiltonian has the following form

$$\bar{H} = uHu^\dagger = \sum_{k,\sigma} \varepsilon_k n_{k\sigma} + \sum_{k,k'} F_{k,k'} n_{k\uparrow} n_{k'\downarrow} + O(U^2) \quad (5)$$

and determine $F_{k,k'}$.








1. From the lecture "Theoretical Solid State Physics" (Theoretische Festkörperphysik), helt by Markus Heyl in WS 22/23
2. Sheet 8, Problem 1: Schrieffer-Wolf transformation of the Hubbard Model
3. Just written down steps done by someone else (Tutor/other pupil, often shared the terms to calculate them all, still time)

# Examples from the Practical Training

## Introduction of the problem

- Goal: produce time evolution of operator

$$\hat{V} = -J \cdot \sum_{\langle l,m \rangle} \left( \hat{c}_l^\dagger \hat{c}_m + \hat{c}_m^\dagger \hat{c}_l + \hat{d}_l^\dagger \hat{d}_m + \hat{d}_m^\dagger \hat{d}_l \right)$$





Examples from the Practical Training
Introduction of the problem
Goal: produce time evolution of operator
$\hat{V} = -J \cdot \sum_{\langle l,m \rangle} \left( \hat{c}_l^\dagger \hat{c}_m + \hat{c}_m^\dagger \hat{c}_l + \hat{d}_l^\dagger \hat{d}_m + \hat{d}_m^\dagger \hat{d}_l \right)$

1. Want to calculate the Time-Evolution of the operator V in the interaction Picture
2. Basically comes down to inserting definitions (however already derived from MM)
3. Distributing and re-ordering is a painful, time-intensive process
4. How to time-evolve operator in the Interaction Picture $\{\cdot\}(t) = e^{iH_0 t} \{\cdot\} e^{-iH_0 t}$

# Examples from the Practical Training

## Introduction of the problem

- Goal: produce time evolution of operator
- Uses calculation in Interaction Picture

$$\hat{\mathrm{V}} = -J \cdot \sum_{\langle l,m\rangle} \left( \hat{\mathrm{c}}_l^\dagger \hat{\mathrm{c}}_m + \hat{\mathrm{c}}_m^\dagger \hat{\mathrm{c}}_l + \hat{\mathrm{d}}_l^\dagger \hat{\mathrm{d}}_m + \hat{\mathrm{d}}_m^\dagger \hat{\mathrm{d}}_l \right)$$

$$\hat{\mathrm{c}}_m^{\dagger\mathrm{I}}(t) = e^{i\cdot\varepsilon_m\cdot t} \left(1 + \left(e^{i\cdot U\cdot t} - 1\right) \hat{\mathrm{d}}_m^{\dagger\mathrm{S}} \hat{\mathrm{d}}_m^{\mathrm{S}} \right) \hat{\mathrm{c}}_m^{\dagger\mathrm{S}}$$

$$\hat{\mathrm{c}}_m^{\mathrm{I}}(t) = e^{-i\cdot\varepsilon_m\cdot t} \left(1 + \left(e^{-i\cdot U\cdot t} - 1\right) \hat{\mathrm{d}}_m^{\dagger\mathrm{S}} \hat{\mathrm{d}}_m^{\mathrm{S}} \right) \hat{\mathrm{c}}_m^{\mathrm{S}}$$

$$\hat{\mathrm{d}}_m^{\dagger\mathrm{I}}(t) = e^{i\cdot\varepsilon_m\cdot t} \left(1 + \left(e^{i\cdot U\cdot t} - 1\right) \hat{\mathrm{c}}_m^{\dagger\mathrm{S}} \hat{\mathrm{c}}_m^{\mathrm{S}} \right) \hat{\mathrm{d}}_m^{\dagger\mathrm{S}}$$

$$\hat{\mathrm{d}}_m^{\mathrm{I}}(t) = e^{-i\cdot\varepsilon_m\cdot t} \left(1 + \left(e^{-i\cdot U\cdot t} - 1\right) \hat{\mathrm{c}}_m^{\dagger\mathrm{S}} \hat{\mathrm{c}}_m^{\mathrm{S}} \right) \hat{\mathrm{d}}_m^{\mathrm{S}}$$





1. Want to calculate the Time-Evolution of the operator V in the interaction Picture
2. Basically comes down to inserting definitions (however already derived from MM)
3. Distributing and re-ordering is a painful, time-intensive process
4. How to time-evolve operator in the Interaction Picture $\{\cdot\}(t) = e^{iH_0 t}\{\cdot\}\, e^{-iH_0 t}$

# Examples from the Practical Training

## Introduction of the problem

- Goal: produce time evolution of operator
- Uses calculation in Interaction Picture
- Workflow:
  - Inserting Definitions
  - Expanding
  - Ordering
  - Rewriting efficiently

$$\hat{\mathrm{V}} = -J \cdot \sum_{\langle l,m \rangle} \left( \hat{\mathrm{c}}_l^\dagger \hat{\mathrm{c}}_m + \hat{\mathrm{c}}_m^\dagger \hat{\mathrm{c}}_l + \hat{\mathrm{d}}_l^\dagger \hat{\mathrm{d}}_m + \hat{\mathrm{d}}_m^\dagger \hat{\mathrm{d}}_l \right)$$

$$\hat{\mathrm{c}}_m^{\dagger\mathrm{I}}(t) = e^{i\cdot\varepsilon_m\cdot t} \left( 1 + \left( e^{i\cdot U\cdot t} - 1 \right) \hat{\mathrm{d}}_m^{\dagger\mathrm{S}} \hat{\mathrm{d}}_m^{\mathrm{S}} \right) \hat{\mathrm{c}}_m^{\dagger\mathrm{S}}$$

$$\hat{\mathrm{c}}_m^{\mathrm{I}}(t) = e^{-i\cdot\varepsilon_m\cdot t} \left( 1 + \left( e^{-i\cdot U\cdot t} - 1 \right) \hat{\mathrm{d}}_m^{\dagger\mathrm{S}} \hat{\mathrm{d}}_m^{\mathrm{S}} \right) \hat{\mathrm{c}}_m^{\mathrm{S}}$$

$$\hat{\mathrm{d}}_m^{\dagger\mathrm{I}}(t) = e^{i\cdot\varepsilon_m\cdot t} \left( 1 + \left( e^{i\cdot U\cdot t} - 1 \right) \hat{\mathrm{c}}_m^{\dagger\mathrm{S}} \hat{\mathrm{c}}_m^{\mathrm{S}} \right) \hat{\mathrm{d}}_m^{\dagger\mathrm{S}}$$

$$\hat{\mathrm{d}}_m^{\mathrm{I}}(t) = e^{-i\cdot\varepsilon_m\cdot t} \left( 1 + \left( e^{-i\cdot U\cdot t} - 1 \right) \hat{\mathrm{c}}_m^{\dagger\mathrm{S}} \hat{\mathrm{c}}_m^{\mathrm{S}} \right) \hat{\mathrm{d}}_m^{\mathrm{S}}$$





1. Want to calculate the Time-Evolution of the operator V in the interaction Picture
2. Basically comes down to inserting definitions (however already derived from MM)
3. Distributing and re-ordering is a painful, time-intensive process
4. How to time-evolve operator in the Interaction Picture $\{\cdot\}(t) = e^{iH_0 t}\{\cdot\}e^{-iH_0 t}$

# Paper-like writing on digital devices

## Solutions: Off the shelf

- Can give benefits that paper lacks
  - Copy/Paste
  - Collaborate
  - Convert to text
  - Searching and ordering large amounts of notes





Paper-like writing on digital devices
Solutions: Off the shelf
Can give benefits that paper lacks
Copy/Paste
Collaborate
Convert to text
Searching and ordering large amounts of notes

1. Taking notes digially can have many benefits
2. OneNote, Xournalpp, Notion and many more examples
3. Still lacks "true" automatization
4. Great danger of Vendor-Locking, losing access
   - Cloud-Driven offers do not work without internet
   - Often times even tied to the device (Apple-Notes I-pad, personal experience with one-note)
   - Countless stories about providers ending support, users getting blocked and more
5. Conclusion: very great choice to boost productivity. But TIP: archive everything as pdf if you ever plan to use it later. It may save you weeks of work/complete loss of data.

# Paper-like writing on digital devices

## Solutions: Off the shelf

- Can give benefits that paper lacks
  - Copy/Paste
  - Collaborate
  - Convert to text
  - Searching and ordering large amounts of notes
- HOWEVER
  - Still no proper automatization
  - Easily access is lost when using proprietary technology
  - "Vendor-Locking"





Paper-like writing on digital devices
Solutions: Off the shelf
Can give benefits that paper lacks
Copy/Paste
Collaborate
Convert to text
Searching and ordering large amounts of notes
HOWEVER
Still no proper automatization
Easily access is lost when using proprietary technology
"Vendor-Locking"

1. Taking notes digially can have many benefits
2. OneNote, Xournalpp, Notion and many more examples
3. Still lacks "true" automatization
4. Great danger of Vendor-Locking, losing access
   - Cloud-Driven offers do not work without internet
   - Often times even tied to the device (Apple-Notes I-pad, personal experience with one-note)
   - Countless stories about providers ending support, users getting blocked and more
5. Conclusion: very great choice to boost productivity. But TIP: archive everything as pdf if you ever plan to use it later. It may save you weeks of work/complete loss of data.

# Paper-like writing on digital devices

## Solutions: Off the shelf

- Can give benefits that paper lacks
  - Copy/Paste
  - Collaborate
  - Convert to text
  - Searching and ordering large amounts of notes
- HOWEVER
  - Still no proper automatization
  - Easily access is lost when using proprietary technology
  - "Vendor-Locking"

**TIP:**

Always export to a standard format (like pdf) for manual backup!





Paper-like writing on digital devices
Solutions: Off the shelf
Can give benefits that paper lacks
Copy/Paste
Collaborate
Convert to text
Searching and ordering large amounts of notes
HOWEVER
Still no proper automatization
Easily access is lost when using proprietary technology
"Vendor-Locking"
TIP:
Always export to a standard format (like pdf) for manual backup!

1. Taking notes digially can have many benefits
2. OneNote, Xournalpp, Notion and many more examples
3. Still lacks "true" automatization
4. Great danger of Vendor-Locking, losing access
   - Cloud-Driven offers do not work without internet
   - Often times even tied to the device (Apple-Notes I-pad, personal experience with one-note)
   - Countless stories about providers ending support, users getting blocked and more
5. Conclusion: very great choice to boost productivity. But TIP: archive everything as pdf if you ever plan to use it later. It may save you weeks of work/complete loss of data.

## Computer-Algebra Systems

### Solutions: Off the shelf

- Proprietary Software (Often payed):

- Standalone Online Calculators





Computer-Algebra Systems
Solutions: Off the shelf
Proprietary Software (Often payed):
Standalone Online Calculators

1. There exist already countless possible attempts at solving the worlds problems
2. All of them are great at what they are designed for, I will not attempt competing with any of the m in terms of any metric except maybe usability (and probably still fail)
3. Chat-GPT is included in the list, because it can do many calculations by now, however is extremely unreliable
4. Central, reoccuring Problems (multiple ore some for everything)
   - Paid access
   - Non-reliable
   - Difficult to "cite" / use in paper/thesis
   - Limited in scope, because software is very spezialized to the task
   - Require training/ prior (export) knowlede to properly take advantage of

# Computer-Algebra Systems

## Solutions: Off the shelf

- Proprietary Software (Often payed):
  - Wolfram-Alpha
  - Mathematica
  - Chat-GPT
  - ...
- Standalone Online Calculators





Computer-Algebra Systems
Solutions: Off the shelf
- Proprietary Software (Often payed):
  - Wolfram-Alpha
  - Mathematica
  - Chat-GPT
  - ...
- Standalone Online Calculators

1. There exist already countless possible attempts at solving the worlds problems
2. All of them are great at what they are designed for, I will not attempt competing with any of the m in terms of any metric except maybe usability (and probably still fail)
3. Chat-GPT is included in the list, because it can do many calculations by now, however is extremely unreliable
4. Central, reoccuring Problems (multiple ore some for everything)
   - Paid access
   - Non-reliable
   - Difficult to "cite" / use in paper/thesis
   - Limited in scope, because software is very spezialized to the task
   - Require training/ prior (export) knowlede to properly take advantage of

# Computer-Algebra Systems

## Solutions: Off the shelf

- Proprietary Software (Often payed):
  - Wolfram-Alpha
  - Mathematica
  - Chat-GPT
  - ...
- Standalone Online Calculators
  - Integralrechner.de (Integral-Calculator)
  - Matrix-Calculators
  - ...





Computer-Algebra Systems
Solutions: Off the shelf
Proprietary Software (Often payed):
Wolfram-Alpha
Mathematica
Chat-GPT
...
Standalone Online Calculators
Integralrechner.de (Integral-Calculator)
Matrix-Calculators
...

1. There exist already countless possible attempts at solving the worlds problems
2. All of them are great at what they are designed for, I will not attempt competing with any of the m in terms of any metric except maybe usability (and probably still fail)
3. Chat-GPT is included in the list, because it can do many calculations by now, however is extremely unreliable
4. Central, reoccuring Problems (multiple ore some for everything)
   - Paid access
   - Non-reliable
   - Difficult to "cite"/ use in paper/thesis
   - Limited in scope, because software is very spezialized to the task
   - Require training/ prior (export) knowlede to properly take advantage of

## Computer-Algebra Systems

### Solutions: Off the shelf

- Proprietary Software (Often payed):
  - Wolfram-Alpha
  - Mathematica
  - Chat-GPT
  - ...
- Standalone Online Calculators
  - Integralrechner.de (Integral-Calculator)
  - Matrix-Calculators
  - ...
- ... and many more





Computer-Algebra Systems
Solutions: Off the shelf
Proprietary Software (Often payed):
Wolfram-Alpha
Mathematica
Chat-GPT
...
Standalone Online Calculators
Integralrechner.de (Integral-Calculator)
Matrix-Calculators
...
... and many more

1. There exist already countless possible attempts at solving the worlds problems
2. All of them are great at what they are designed for, I will not attempt competing with any of the m in terms of any metric except maybe usability (and probably still fail)
3. Chat-GPT is included in the list, because it can do many calculations by now, however is extremely unreliable
4. Central, reoccuring Problems (multiple ore some for everything)
   - Paid access
   - Non-reliable
   - Difficult to "cite" / use in paper/thesis
   - Limited in scope, because software is very spezialized to the task
   - Require training/ prior (export) knowlede to properly take advantage of

# What is Math-Manipulator?

## Math-Manipulator - What did I do?

```
sum((n = 0); 100; int(-inf; inf; (123+(A*4)/100); x))
```

$$\sum_{n=0}^{100} \int_{-\infty}^{\infty} \left(123 + \boxed{\frac{(A \cdot 4)}{100}}\right) \mathrm{d}x$$

Sel. Parent | Sel. First Child | Sel. Previous Sibling | Sel. Next Sibling

Replace | Replace All Equal | Replace (With Export) | Define Variable | Commute with subsequent | Fold Numbers | Cleanup Terms | Pull Out Minus | Reduce | Rename With Swap | Pack Same Variables | Replace All With

Selected Export to Clipboard | Show Export Structure (UUIDs) | Show Export Structure | Show Latex

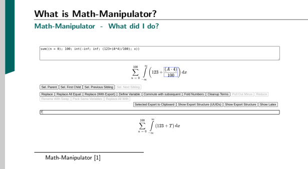
```
T
```

$$\sum_{n=0}^{100} \int_{-\infty}^{\infty} (123 + T) \, \mathrm{d}x$$

Math-Manipulator [1]







1. I wrote a custom tool that should help with the process of doing math paper-like, but with the help of computers
2. Features/Properties
   - Relatively smart initial text-input
     - Omit implied multiplication
     - Omit implied zeros
     - Parameter autoboxing
     - Bracket culling
     - $\rightarrow$ Way less stringent as e.g. input to many Computer-Algebra-Systems requires
   - Once initial input is set, operations with mouse
     - Rename, Pack, Move elements with mouse interactions
     - Execute operator-specific operations (distribute, delta/qm-op stuff)
   - Many pre-defined Functions/Symbols
   - Automatic Macro and Variable system
   - In-Browser Help with interactive Playground

# In the Browser

## Math-Manipulator - How can you use it

- Available open-source on GitHub
- Self-hosting possible
- Hosted version available in every modern browser





1. Source code available
2. (Relatively) easily extensible, tutorial and issues available
3. Instantly usable, offline and online

In the Browser
Math-Manipulator - How can you use it
- Available open-source on GitHub
- Self-hosting possible
- Hosted version available in every modern browser

# In the Browser

## Math-Manipulator - How can you use it

- Available open-source on GitHub
- Self-hosting possible
- Hosted version available in every modern browser

**Available here**

https://jonas-kell.github.io/math-manipulator/





1. Source code available
2. (Relatively) easily extensible, tutorial and issues available
3. Instantly usable, offline and online

# Usage as VS-Code Extension

## Math-Manipulator - How can you use it

- Offline-Use after installed once
- Store progress in files
- Directly integrated in powerful IDE, and with that version control





Usage as VS-Code Extension
Math-Manipulator - How can you use it
- Offline-Use after installed once
- Store progress in files
- Directly integrated in powerful IDE, and with that version control

1. For more serious use, integrated in the code-editor VS-Code as an Extension
2. Allows for all features, but includes
   - Offline use out of the box
   - Storage of files (with that version control)
   - Multiple parallel projects
3. Has already taken entry into my personal workflow by now
   - Probably I will never make back my WHOLE time-investment
   - But I already saved hours upon hours when I noticed sign mistakes, symmetry issues or similar in calculations I did with it

# Usage as VS-Code Extension

## Math-Manipulator - How can you use it

- Offline-Use after installed once
- Store progress in files
- Directly integrated in powerful IDE, and with that version control

**Note**

Do not allow a spellchecker to run on the file, it will break everything performance-wise





Usage as VS-Code Extension
Math-Manipulator - How can you use it
Offline-Use after installed once
Store progress in files
Directly integrated in powerful IDE, and with that version control
Note
Do not allow a spellchecker to run on the file, it will break everything performance-wise

1. For more serious use, integrated in the code-editor VS-Code as an Extension
2. Allows for all features, but includes
   - Offline use out of the box
   - Storage of files (with that version control)
   - Multiple parallel projects
3. Has already taken entry into my personal workflow by now
   - Probably I will never make back my WHOLE time-investment
   - But I already saved hours upon hours when I noticed sign mistakes, symmetry issues or similar in calculations I did with it

# Demo Problems from before

## Math-Manipulator - Interactive Demonstration

$$= \epsilon\, [c_1^\dagger c_2, c_1^\dagger c_1] - \epsilon\, [c_2^\dagger c_1 - c_1^\dagger c_1] + \Delta\, [c_1^\dagger c_2, c_2^\dagger c_1] - \Delta\, [c_2^\dagger c_1, c_1^\dagger c_2]$$

$$= \epsilon\, (c_1^\dagger c_2 - c_2^\dagger c_1) + 2\Delta\, (c_1^\dagger c_1 - c_2^\dagger c_2)$$

$$[c_{k_1\uparrow}^\dagger c_{k_2\uparrow} c_{k_3\downarrow}^\dagger c_{k_4\downarrow}, \sum_{k\sigma} \epsilon_k n_{k\sigma}]$$

$$-(\epsilon_{k_1} - \epsilon_{k_2} + \epsilon_{k_3} - \epsilon_{k_4})\, c_{k_1\uparrow}^\dagger c_{k_2\uparrow} c_{k_3\downarrow}^\dagger c_{k_4\downarrow}$$





1. Go to Demo in VS-Code folder math-manipulator-calculations in the presentation subfolder
2. Think of disabling spellchecker
3. Think to first split deltas before evaluating !!!!!!
4. Macros and other optimizations are possible, but not really necessary
5. Input for simple demo: distribute, cleanup, order op string,
   `(x+y)(4*x -z+y )(-z + 1+ x)`
6. Input for simple operators:
   `fc("" 1)fa("" 1) ;|; fc("" 1)fa("" m) ;|; fc("" 1)fa("" 2) ;|; fa("" 2)fc("" 2);|; fc("" 1)fa("" 1)fc("" 3)fa("" 3)fa("" 2)fc("" 1)`

# Usage for the Practical Training

## Math-Manipulator - Interactive Demonstration

$$\hat{\mathrm{V}}^{\mathrm{I}}(t) = \left\{\hat{\mathrm{V}}^{\mathrm{S}}\right\}(t) \overset{2.11}{=} -J \cdot \sum_{[l,m]} \left\{\left(\hat{\mathrm{c}}_l^{\dagger\mathrm{S}}\hat{\mathrm{c}}_m^{\mathrm{S}} + \hat{\mathrm{d}}_l^{\dagger\mathrm{S}}\hat{\mathrm{d}}_m^{\mathrm{S}}\right)\right\}(t)$$

$$= -J \cdot \sum_{[l,m]} \left(\hat{\mathrm{c}}_l^{\dagger\mathrm{I}}(t)\hat{\mathrm{c}}_m^{\mathrm{I}}(t) + \hat{\mathrm{d}}_l^{\dagger\mathrm{I}}(t)\hat{\mathrm{d}}_m^{\mathrm{I}}(t)\right)$$

$$\overset{\mathrm{MM}}{=} -J \cdot \sum_{[l,m]} \Big[e^{i\cdot(\varepsilon_m-\varepsilon_l)\cdot t} \cdot \hat{\mathrm{V}}_{\text{Part A}}(l,m) + e^{i\cdot(\varepsilon_m-\varepsilon_l+U)\cdot t} \cdot \hat{\mathrm{V}}_{\text{Part B}}(l,m) + e^{i\cdot(\varepsilon_m-\varepsilon_l-U)\cdot t} \cdot \hat{\mathrm{V}}_{\text{Part C}}(l,m)\Big]$$





1. Go to Demo in VS-Code submodule mm-calculations.
2. Think of disabling spellchecker
3. Show process of how V is calculated
4. Story: at the end at first result was no longer symmetrical in up and down, but it should have been. Mistake was in the very first input. But because of the upfornt configuration (and in my case programming) effort, re-calculating the endresult took 20min instead of one week

# Usage for the Practical Training

## Math-Manipulator - Interactive Demonstration

$$
\begin{aligned}
\hat{\mathrm{V}}^{\mathrm{I}}(t) &= \left\{\hat{\mathrm{V}}^{\mathrm{S}}\right\}(t) \overset{2.11}{=} -J \cdot \sum_{[l,m]} \left\{\left(\hat{\mathrm{c}}_l^{\dagger\mathrm{S}}\hat{\mathrm{c}}_m^{\mathrm{S}} + \hat{\mathrm{d}}_l^{\dagger\mathrm{S}}\hat{\mathrm{d}}_m^{\mathrm{S}}\right)\right\}(t) \\
&= -J \cdot \sum_{[l,m]} \left(\hat{\mathrm{c}}_l^{\dagger\mathrm{I}}(t)\hat{\mathrm{c}}_m^{\mathrm{I}}(t) + \hat{\mathrm{d}}_l^{\dagger\mathrm{I}}(t)\hat{\mathrm{d}}_m^{\mathrm{I}}(t)\right) \\
&\overset{\mathrm{MM}}{=} -J \cdot \sum_{[l,m]} \Big[e^{i\cdot(\varepsilon_m-\varepsilon_l)\cdot t} \cdot \hat{\mathrm{V}}_{\text{Part A}}(l,m) + \\
&\qquad e^{i\cdot(\varepsilon_m-\varepsilon_l+U)\cdot t} \cdot \hat{\mathrm{V}}_{\text{Part B}}(l,m) + e^{i\cdot(\varepsilon_m-\varepsilon_l-U)\cdot t} \cdot \hat{\mathrm{V}}_{\text{Part C}}(l,m)\Big]
\end{aligned}
$$

$$
\begin{aligned}
\hat{\mathrm{V}}_{\text{Part A}}(l,m) &\overset{\mathrm{MM}}{=} \left(5 \cdot \hat{\mathrm{c}}_l^{\mathrm{S}}\hat{\mathrm{c}}_m^{\dagger\mathrm{S}}\right) + \left(5 \cdot \hat{\mathrm{d}}_l^{\mathrm{S}}\hat{\mathrm{d}}_m^{\dagger\mathrm{S}}\right) + \left(2 \cdot \hat{\mathrm{c}}_l^{\mathrm{S}}\hat{\mathrm{c}}_m^{\mathrm{S}}\hat{\mathrm{c}}_l^{\dagger\mathrm{S}}\hat{\mathrm{c}}_m^{\dagger\mathrm{S}}\hat{\mathrm{d}}_l^{\mathrm{S}}\hat{\mathrm{d}}_m^{\dagger\mathrm{S}}\right) + \left(2 \cdot \hat{\mathrm{c}}_l^{\mathrm{S}}\hat{\mathrm{c}}_m^{\dagger\mathrm{S}}\hat{\mathrm{d}}_l^{\mathrm{S}}\hat{\mathrm{d}}_m^{\mathrm{S}}\hat{\mathrm{d}}_l^{\dagger\mathrm{S}}\hat{\mathrm{d}}_m^{\dagger\mathrm{S}}\right) + \\
&+ \left(-3 \cdot \hat{\mathrm{c}}_l^{\mathrm{S}}\hat{\mathrm{c}}_l^{\dagger\mathrm{S}}\hat{\mathrm{d}}_l^{\mathrm{S}}\hat{\mathrm{d}}_m^{\dagger\mathrm{S}}\right) + \left(-3 \cdot \hat{\mathrm{c}}_m^{\mathrm{S}}\hat{\mathrm{c}}_m^{\dagger\mathrm{S}}\hat{\mathrm{d}}_l^{\mathrm{S}}\hat{\mathrm{d}}_m^{\dagger\mathrm{S}}\right) + \left(-3 \cdot \hat{\mathrm{c}}_l^{\mathrm{S}}\hat{\mathrm{c}}_m^{\dagger\mathrm{S}}\hat{\mathrm{d}}_l^{\mathrm{S}}\hat{\mathrm{d}}_l^{\dagger\mathrm{S}}\right) + \left(-3 \cdot \hat{\mathrm{c}}_l^{\mathrm{S}}\hat{\mathrm{c}}_m^{\dagger\mathrm{S}}\hat{\mathrm{d}}_m^{\mathrm{S}}\hat{\mathrm{d}}_m^{\dagger\mathrm{S}}\right) \\
\hat{\mathrm{V}}_{\text{Part B}}(l,m) &\overset{\mathrm{MM}}{=} \left(-2 \cdot \hat{\mathrm{c}}_l^{\mathrm{S}}\hat{\mathrm{c}}_m^{\dagger\mathrm{S}}\right) + \left(-2 \cdot \hat{\mathrm{d}}_l^{\mathrm{S}}\hat{\mathrm{d}}_m^{\dagger\mathrm{S}}\right) + \left(-1 \cdot \hat{\mathrm{c}}_l^{\mathrm{S}}\hat{\mathrm{c}}_m^{\mathrm{S}}\hat{\mathrm{c}}_l^{\dagger\mathrm{S}}\hat{\mathrm{c}}_m^{\dagger\mathrm{S}}\hat{\mathrm{d}}_l^{\mathrm{S}}\hat{\mathrm{d}}_m^{\dagger\mathrm{S}}\right) + \left(-1 \cdot \hat{\mathrm{c}}_l^{\mathrm{S}}\hat{\mathrm{c}}_m^{\dagger\mathrm{S}}\hat{\mathrm{d}}_l^{\mathrm{S}}\hat{\mathrm{d}}_m^{\mathrm{S}}\hat{\mathrm{d}}_l^{\dagger\mathrm{S}}\hat{\mathrm{d}}_m^{\dagger\mathrm{S}}\right) + \\
&+ \left(1 \cdot \hat{\mathrm{c}}_l^{\mathrm{S}}\hat{\mathrm{c}}_l^{\dagger\mathrm{S}}\hat{\mathrm{d}}_l^{\mathrm{S}}\hat{\mathrm{d}}_m^{\dagger\mathrm{S}}\right) + \left(2 \cdot \hat{\mathrm{c}}_m^{\mathrm{S}}\hat{\mathrm{c}}_m^{\dagger\mathrm{S}}\hat{\mathrm{d}}_l^{\mathrm{S}}\hat{\mathrm{d}}_m^{\dagger\mathrm{S}}\right) + \left(1 \cdot \hat{\mathrm{c}}_l^{\mathrm{S}}\hat{\mathrm{c}}_m^{\dagger\mathrm{S}}\hat{\mathrm{d}}_l^{\mathrm{S}}\hat{\mathrm{d}}_l^{\dagger\mathrm{S}}\right) + \left(2 \cdot \hat{\mathrm{c}}_l^{\mathrm{S}}\hat{\mathrm{c}}_m^{\dagger\mathrm{S}}\hat{\mathrm{d}}_m^{\mathrm{S}}\hat{\mathrm{d}}_m^{\dagger\mathrm{S}}\right) \\
\hat{\mathrm{V}}_{\text{Part C}}(l,m) &\overset{\mathrm{MM}}{=} \left(-2 \cdot \hat{\mathrm{c}}_l^{\mathrm{S}}\hat{\mathrm{c}}_m^{\dagger\mathrm{S}}\right) + \left(-2 \cdot \hat{\mathrm{d}}_l^{\mathrm{S}}\hat{\mathrm{d}}_m^{\dagger\mathrm{S}}\right) + \left(-1 \cdot \hat{\mathrm{c}}_l^{\mathrm{S}}\hat{\mathrm{c}}_m^{\mathrm{S}}\hat{\mathrm{c}}_l^{\dagger\mathrm{S}}\hat{\mathrm{c}}_m^{\dagger\mathrm{S}}\hat{\mathrm{d}}_l^{\mathrm{S}}\hat{\mathrm{d}}_m^{\dagger\mathrm{S}}\right) + \left(-1 \cdot \hat{\mathrm{c}}_l^{\mathrm{S}}\hat{\mathrm{c}}_m^{\dagger\mathrm{S}}\hat{\mathrm{d}}_l^{\mathrm{S}}\hat{\mathrm{d}}_m^{\mathrm{S}}\hat{\mathrm{d}}_l^{\dagger\mathrm{S}}\hat{\mathrm{d}}_m^{\dagger\mathrm{S}}\right) + \\
&+ \left(2 \cdot \hat{\mathrm{c}}_l^{\mathrm{S}}\hat{\mathrm{c}}_l^{\dagger\mathrm{S}}\hat{\mathrm{d}}_l^{\mathrm{S}}\hat{\mathrm{d}}_m^{\dagger\mathrm{S}}\right) + \left(1 \cdot \hat{\mathrm{c}}_m^{\mathrm{S}}\hat{\mathrm{c}}_m^{\dagger\mathrm{S}}\hat{\mathrm{d}}_l^{\mathrm{S}}\hat{\mathrm{d}}_m^{\dagger\mathrm{S}}\right) + \left(2 \cdot \hat{\mathrm{c}}_l^{\mathrm{S}}\hat{\mathrm{c}}_m^{\dagger\mathrm{S}}\hat{\mathrm{d}}_l^{\mathrm{S}}\hat{\mathrm{d}}_l^{\dagger\mathrm{S}}\right) + \left(1 \cdot \hat{\mathrm{c}}_l^{\mathrm{S}}\hat{\mathrm{c}}_m^{\dagger\mathrm{S}}\hat{\mathrm{d}}_m^{\mathrm{S}}\hat{\mathrm{d}}_m^{\dagger\mathrm{S}}\right)
\end{aligned}
$$





1. Go to Demo in VS-Code submodule mm-calculations.
2. Think of disabling spellchecker
3. Show process of how V is calculated
4. Story: at the end at first result was no longer symmetrical in up and down, but it should have been. Mistake was in the very first input. But because of the upfont configuration (and in my case programming) effort, re-calculating the endresult took 20min instead of one week

# Optimization: Difference of V(t) for Hopping

## Custom Python Scripts (SymPy)

- Monte-Carlo sampling requires transition probabilities between "adjacent" states

$$\alpha = \frac{P(\tilde{N},t)}{P(N,t)} = \frac{f(\tilde{N},t)}{f(N,t)} \overset{2.39}{=} \frac{\left|e^{\mathcal{H}_{\text{eff}}(\tilde{N},t)}\right|^2 \left|\Psi_{\tilde{N}}\right|^2}{\left|e^{\mathcal{H}_{\text{eff}}(N,t)}\right|^2 \left|\Psi_N\right|^2}$$
$$= \frac{\left|\Psi_{\tilde{N}}\right|^2}{\left|\Psi_N\right|^2} \frac{e^{\Re\left(\mathcal{H}_{\text{eff}}(\tilde{N},t)\right)+i\Im\left(\mathcal{H}_{\text{eff}}(\tilde{N},t)\right)+\Re\left(\mathcal{H}_{\text{eff}}(\tilde{N},t)\right)-i\Im\left(\mathcal{H}_{\text{eff}}(\tilde{N},t)\right)}}{e^{\Re(\mathcal{H}_{\text{eff}}(N,t))+i\Im(\mathcal{H}_{\text{eff}}(N,t))+\Re(\mathcal{H}_{\text{eff}}(N,t))-i\Im(\mathcal{H}_{\text{eff}}(N,t))}}$$
$$= \frac{\left|\Psi_{\tilde{N}}\right|^2}{\left|\Psi_N\right|^2} e^{2\cdot\Re\left(\mathcal{H}_{\text{eff}}(\tilde{N},t)\right)-2\cdot\Re(\mathcal{H}_{\text{eff}}(N,t))}$$
$$= \frac{\left|\Psi_{\tilde{N}}\right|^2}{\left|\Psi_N\right|^2} e^{2\cdot\Re\left(\mathcal{H}_{\text{eff}}(\tilde{N},t)-\mathcal{H}_{\text{eff}}(N,t)\right)}$$





1. "Adjacent" states because from one Monte-Carlo step to the next we require a small change of one or two particle-sites. Could come from hopping or flipping.
2. The $E_0$ differences are quickly calculated by hand in the report

# Optimization: Difference of V(t) for Hopping

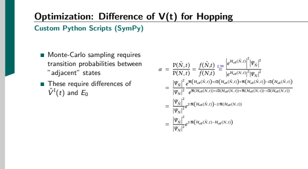
## Custom Python Scripts (SymPy)

- Monte-Carlo sampling requires transition probabilities between "adjacent" states
- These require differences of $\hat{V}^{\mathrm{I}}(t)$ and $E_0$

$$
\begin{aligned}
\alpha &= \frac{\mathrm{P}(\tilde{N},t)}{\mathrm{P}(N,t)} = \frac{f(\tilde{N},t)}{f(N,t)} \overset{2.39}{=} \frac{\left|e^{\mathcal{H}_{\mathrm{eff}}(\tilde{N},t)}\right|^2 \left|\Psi_{\tilde{N}}\right|^2}{\left|e^{\mathcal{H}_{\mathrm{eff}}(N,t)}\right|^2 \left|\Psi_{N}\right|^2} \\
&= \frac{\left|\Psi_{\tilde{N}}\right|^2}{\left|\Psi_{N}\right|^2} \frac{e^{\Re\left(\mathcal{H}_{\mathrm{eff}}(\tilde{N},t)\right)+i\Im\left(\mathcal{H}_{\mathrm{eff}}(\tilde{N},t)\right)+\Re\left(\mathcal{H}_{\mathrm{eff}}(\tilde{N},t)\right)-i\Im\left(\mathcal{H}_{\mathrm{eff}}(\tilde{N},t)\right)}}{e^{\Re(\mathcal{H}_{\mathrm{eff}}(N,t))+i\Im(\mathcal{H}_{\mathrm{eff}}(N,t))+\Re(\mathcal{H}_{\mathrm{eff}}(N,t))-i\Im(\mathcal{H}_{\mathrm{eff}}(N,t))}} \\
&= \frac{\left|\Psi_{\tilde{N}}\right|^2}{\left|\Psi_{N}\right|^2} e^{2\cdot\Re\left(\mathcal{H}_{\mathrm{eff}}(\tilde{N},t)\right)-2\cdot\Re(\mathcal{H}_{\mathrm{eff}}(N,t))} \\
&= \frac{\left|\Psi_{\tilde{N}}\right|^2}{\left|\Psi_{N}\right|^2} e^{2\cdot\Re\left(\mathcal{H}_{\mathrm{eff}}(\tilde{N},t)-\mathcal{H}_{\mathrm{eff}}(N,t)\right)}
\end{aligned}
$$







1. "Adjacent" states because from one Monte-Carlo step to the next we require a small change of one or two particle-sites. Could come from hopping or flipping.
2. The $E_0$ differences are quickly calculated by hand in the report

# Optimization: Difference of V(t) for Hopping

## Custom Python Scripts (SymPy)

- Monte-Carlo sampling requires transition probabilities between "adjacent" states
- These require differences of $\hat{V}^{\mathrm{I}}(t)$ and $E_0$
- Too many terms for simple evaluation by hand
- Intelligent pre-computation required to speed up processing

$$
\begin{aligned}
\alpha &= \frac{\mathrm{P}(\tilde{N},t)}{\mathrm{P}(N,t)} = \frac{f(\tilde{N},t)}{f(N,t)} \overset{2.39}{=} \frac{\left|e^{\mathcal{H}_{\mathrm{eff}}(\tilde{N},t)}\right|^2 \left|\Psi_{\tilde{N}}\right|^2}{\left|e^{\mathcal{H}_{\mathrm{eff}}(N,t)}\right|^2 \left|\Psi_N\right|^2} \\
&= \frac{\left|\Psi_{\tilde{N}}\right|^2}{\left|\Psi_N\right|^2} \frac{e^{\Re\left(\mathcal{H}_{\mathrm{eff}}(\tilde{N},t)\right)+i\Im\left(\mathcal{H}_{\mathrm{eff}}(\tilde{N},t)\right)+\Re\left(\mathcal{H}_{\mathrm{eff}}(\tilde{N},t)\right)-i\Im\left(\mathcal{H}_{\mathrm{eff}}(\tilde{N},t)\right)}}{e^{\Re(\mathcal{H}_{\mathrm{eff}}(N,t))+i\Im(\mathcal{H}_{\mathrm{eff}}(N,t))+\Re(\mathcal{H}_{\mathrm{eff}}(N,t))-i\Im(\mathcal{H}_{\mathrm{eff}}(N,t))}} \\
&= \frac{\left|\Psi_{\tilde{N}}\right|^2}{\left|\Psi_N\right|^2} e^{2\cdot\Re\left(\mathcal{H}_{\mathrm{eff}}(\tilde{N},t)\right)-2\cdot\Re(\mathcal{H}_{\mathrm{eff}}(N,t))} \\
&= \frac{\left|\Psi_{\tilde{N}}\right|^2}{\left|\Psi_N\right|^2} e^{2\cdot\Re\left(\mathcal{H}_{\mathrm{eff}}(\tilde{N},t)-\mathcal{H}_{\mathrm{eff}}(N,t)\right)}
\end{aligned}
$$





1. "Adjacent" states because from one Monte-Carlo step to the next we require a small change of one or two particle-sites. Could come from hopping or flipping.
2. The $E_0$ differences are quickly calculated by hand in the report

# The process: Difference of V(t) for Hopping

## Custom Python Scripts (SymPy)

$$
\begin{aligned}
E_0(N) - E_0(\tilde{N}) &= U \sum_l n_{l,\downarrow} n_{l,\uparrow} - U \sum_l \tilde{n}_{l,\downarrow} \tilde{n}_{l,\uparrow} + \sum_{l,\sigma} \varepsilon_l n_{l,\sigma} - \sum_{l,\sigma} \varepsilon_l \tilde{n}_{l,\sigma} \\
&\overset{2.45}{=} \left(\varepsilon_i - \varepsilon_j\right) \left(n_{i,\sigma_i} - n_{j,\sigma_j}\right) + U \cdot \begin{cases} \left(n_{i,\uparrow} - n_{j,\uparrow}\right)\left(n_{i,\downarrow} - n_{j,\downarrow}\right) & : \sigma_i = \sigma_j \\ \left(n_{i,\uparrow} - n_{j,\downarrow}\right)\left(n_{i,\downarrow} - n_{j,\uparrow}\right) & : \sigma_i \neq \sigma_j \end{cases} \\
&= \left(\varepsilon_i - \varepsilon_j\right) \left(n_{i,\sigma_i} - n_{j,\sigma_j}\right) + U \left(n_{i,\sigma_i} - n_{j,\sigma_j}\right) \left(n_{i,\overline{\sigma_i}} - n_{j,\overline{\sigma_j}}\right)
\end{aligned}
$$

- Setup difference





1. The $V$ differences are $3 \cdot 8 \cdot = 32 \cdot 4 = 96 \cdot 2 = 192$ terms
2. The process is relatively straight forward:
   - write down the terms
   - replace
   - simplify difference
3. Still quite complicated and a lot of work, but:
   - way faster to write bacuse non-repetetive logic
   - verifiably correct calculations
   - multiple times re-done on changes to underlying code, each time only took seconds
   - sometimes brute-force NEEDS to work, before optimization can be found

## The process: Difference of V(t) for Hopping

### Custom Python Scripts (SymPy)

$$
\begin{aligned}
E_0(N) - E_0(\tilde{N}) &= U \sum_l n_{l,\downarrow} n_{l,\uparrow} - U \sum_l \tilde{n}_{l,\downarrow} \tilde{n}_{l,\uparrow} + \sum_{l,\sigma} \varepsilon_l n_{l,\sigma} - \sum_{l,\sigma} \varepsilon_l \tilde{n}_{l,\sigma} \\
&\overset{2.45}{=} \left(\varepsilon_i - \varepsilon_j\right)\left(n_{i,\sigma_i} - n_{j,\sigma_j}\right) + U \cdot \begin{cases} \left(n_{i,\uparrow} - n_{j,\uparrow}\right)\left(n_{i,\downarrow} - n_{j,\downarrow}\right) & : \sigma_i = \sigma_j \\ \left(n_{i,\uparrow} - n_{j,\downarrow}\right)\left(n_{i,\downarrow} - n_{j,\uparrow}\right) & : \sigma_i \neq \sigma_j \end{cases} \qquad \tilde{n}_{l,\sigma} = \begin{cases} n_{i,\sigma_i} & : l = j \wedge \sigma = \sigma_j \\ n_{j,\sigma_j} & : l = i \wedge \sigma = \sigma_i \\ n_{l,\sigma} & : \text{else} \end{cases} \\
&= \left(\varepsilon_i - \varepsilon_j\right)\left(n_{i,\sigma_i} - n_{j,\sigma_j}\right) + U\left(n_{i,\sigma_i} - n_{j,\sigma_j}\right)\left(n_{i,\overline{\sigma_i}} - n_{j,\overline{\sigma_j}}\right)
\end{aligned}
$$

- Setup difference
- Replace one by appropriate new occupation





The process: Difference of V(t) for Hopping
Custom Python Scripts (SymPy)
Setup difference
Replace one by appropriate new occupation

1. The $V$ differences are $3 \cdot 8 \cdot = 32 \cdot 4 = 96 \cdot 2 = 192$ terms
2. The process is relatively straight forward:
   - write down the terms
   - replace
   - simplify difference
3. Still quite complicated and a lot of work, but:
   - way faster to write bacuse non-repetetive logic
   - verifiably correct calculations
   - multiple times re-done on changes to underlying code, each time only took seconds
   - sometimes brute-force NEEDS to work, before optimization can be found

# The process: Difference of V(t) for Hopping

## Custom Python Scripts (SymPy)

$$
\begin{aligned}
E_0(N) - E_0(\tilde{N}) &= U \sum_{l} n_{l,\downarrow} n_{l,\uparrow} - U \sum_{l} \tilde{n}_{l,\downarrow} \tilde{n}_{l,\uparrow} + \sum_{l,\sigma} \varepsilon_l n_{l,\sigma} - \sum_{l,\sigma} \varepsilon_l \tilde{n}_{l,\sigma} \\
&\overset{2.45}{=} \left(\varepsilon_i - \varepsilon_j\right) \left(n_{i,\sigma_i} - n_{j,\sigma_j}\right) + U \cdot \begin{cases} \left(n_{i,\uparrow} - n_{j,\uparrow}\right) \left(n_{i,\downarrow} - n_{j,\downarrow}\right) & : \sigma_i = \sigma_j \\ \left(n_{i,\uparrow} - n_{j,\downarrow}\right) \left(n_{i,\downarrow} - n_{j,\uparrow}\right) & : \sigma_i \neq \sigma_j \end{cases} \\
&= \left(\varepsilon_i - \varepsilon_j\right) \left(n_{i,\sigma_i} - n_{j,\sigma_j}\right) + U \left(n_{i,\sigma_i} - n_{j,\sigma_j}\right) \left(n_{i,\overline{\sigma_i}} - n_{j,\overline{\sigma_j}}\right)
\end{aligned}
$$

$$
\tilde{n}_{l,\sigma} = \begin{cases} n_{i,\sigma_i} & : l = j \wedge \sigma = \sigma_j \\ n_{j,\sigma_j} & : l = i \wedge \sigma = \sigma_i \\ n_{l,\sigma} & : \text{else} \end{cases}
$$

- Setup difference
- Replace one by appropriate new occupation
- Simplify as much as possible
- Sum over all possible neighbor combinations





1. The $V$ differences are $3 \cdot 8 \cdot = 32 \cdot 4 = 96 \cdot 2 = 192$ terms
2. The process is relatively straight forward:
   - write down the terms
   - replace
   - simplify difference
3. Still quite complicated and a lot of work, but:
   - way faster to write bacuse non-repetetive logic
   - verifiably correct calculations
   - multiple times re-done on changes to underlying code, each time only took seconds
   - sometimes brute-force NEEDS to work, before optimization can be found

# The process: Difference of V(t) for Hopping

## Custom Python Scripts (SymPy)

$$
\begin{aligned}
E_0(N) - E_0(\tilde{N}) &= U\sum_{l} n_{l,\downarrow}n_{l,\uparrow} - U\sum_{l}\tilde{n}_{l,\downarrow}\tilde{n}_{l,\uparrow} + \sum_{l,\sigma}\varepsilon_l n_{l,\sigma} - \sum_{l,\sigma}\varepsilon_l \tilde{n}_{l,\sigma} \\
&\overset{2.45}{=} \left(\varepsilon_i - \varepsilon_j\right)\left(n_{i,\sigma_i} - n_{j,\sigma_j}\right) + U\cdot\begin{cases}\left(n_{i,\uparrow} - n_{j,\uparrow}\right)\left(n_{i,\downarrow} - n_{j,\downarrow}\right) & : \sigma_i = \sigma_j \\ \left(n_{i,\uparrow} - n_{j,\downarrow}\right)\left(n_{i,\downarrow} - n_{j,\uparrow}\right) & : \sigma_i \neq \sigma_j\end{cases} \qquad \tilde{n}_{l,\sigma} = \begin{cases} n_{i,\sigma_i} & : l = j \wedge \sigma = \sigma_j \\ n_{j,\sigma_j} & : l = i \wedge \sigma = \sigma_i \\ n_{l,\sigma} & : \text{else}\end{cases} \\
&= \left(\varepsilon_i - \varepsilon_j\right)\left(n_{i,\sigma_i} - n_{j,\sigma_j}\right) + U\left(n_{i,\sigma_i} - n_{j,\sigma_j}\right)\left(n_{i,\overline{\sigma_i}} - n_{j,\overline{\sigma_j}}\right)
\end{aligned}
$$

- Setup difference
- Replace one by appropriate new occupation
- Simplify as much as possible
- Sum over all possible neighbor combinations

**Problem:** number of terms/combinations: $3 \cdot 8 \cdot 4 \cdot 2 = 192$





1. The $V$ differences are $3 \cdot 8 \cdot = 32 \cdot 4 = 96 \cdot 2 = 192$ terms
2. The process is relatively straight forward:
   - write down the terms
   - replace
   - simplify difference
3. Still quite complicated and a lot of work, but:
   - way faster to write bacuse non-repetetive logic
   - verifiably correct calculations
   - multiple times re-done on changes to underlying code, each time only took seconds
   - sometimes brute-force NEEDS to work, before optimization can be found

# Script-Generation: Difference of V(t) for Hopping

## Custom Python Scripts (SymPy)

**Generating Script**

simplificationtermhelper.py

**Generated Script**

analyticalcalcfunctions.py





Script-Generation: Difference of V(t) for Hopping
Custom Python Scripts (SymPy)
Generating Script
simplificationtermhelper.py
Generated Script
analyticalcalcfunctions.py

1. `simplificationtermhelper.py` script to generate the script
2. `analyticalcalcfunctions.py` script is generated and used in the code
3. Generated file: 1556 lines, script that generates: 534 lines
4. Show the script that generates/is generated → simplification with Sympy
5. Parallel to computer-science practice "Vendored" code

# Beamer: LaTeX way of writing Presentations

## Presentations? - Custom Beamer Template

- Write presentations like your papers/thesis in LaTeX

Beamer [2] Template [3]





Beamer: LaTeX way of writing Presentations
Presentations? - Custom Beamer Template
Write presentations like your papers/thesis in LaTeX
Beamer [2] Template [3]

1. Re-use tooling, editor, setup (even this on overleaf alone much better that shared powerpoint presentation)
2. This presentation, report and thesis all live in one git-repository and share resources
3. Everything I ever work on is hosted in Git.
   - Without Version Control it is no longer possible for me to work effectively
   - Every workflow is massively ensured by it
   - Also for single-person work, but built in collaboration, synchronization and backup

## Beamer: LaTeX way of writing Presentations

### Presentations? - Custom Beamer Template

- Write presentations like your papers/thesis in LaTeX
- Reuse formulas/images/code/sources

Beamer [2] Template [3]





Beamer: LaTeX way of writing Presentations
Presentations? - Custom Beamer Template
Write presentations like your papers/thesis in LaTeX
Reuse formulas/images/code/sources
Beamer [2] Template [3]

1. Re-use tooling, editor, setup (even this on overleaf alone much better that shared powerpoint presentation)
2. This presentation, report and thesis all live in one git-repository and share resources
3. Everything I ever work on is hosted in Git.
   - Without Version Control it is no longer possible for me to work effectively
   - Every workflow is massively ensured by it
   - Also for single-person work, but built in collaboration, synchronization and backup

# Beamer: LaTeX way of writing Presentations

## Presentations? - Custom Beamer Template

- Write presentations like your papers/thesis in LaTeX
- Reuse formulas/images/code/sources
- Consistent style & references

Beamer [2] Template [3]





Beamer: LaTeX way of writing Presentations
Presentations? - Custom Beamer Template
Write presentations like your papers/thesis in LaTeX
Reuse formulas/images/code/sources
Consistent style & references
Beamer [2] Template [3]

1. Re-use tooling, editor, setup (even this on overleaf alone much better that shared powerpoint presentation)
2. This presentation, report and thesis all live in one git-repository and share resources
3. Everything I ever work on is hosted in Git.
   - Without Version Control it is no longer possible for me to work effectively
   - Every workflow is massively ensured by it
   - Also for single-person work, but built in collaboration, synchronization and backup

# Beamer: LaTeX way of writing Presentations

## Presentations? - Custom Beamer Template

- Write presentations like your papers/thesis in LaTeX
- Reuse formulas/images/code/sources
- Consistent style & references
- Version control

Beamer [2] Template [3]





Beamer: LaTeX way of writing Presentations
Presentations? - Custom Beamer Template
Write presentations like your papers/thesis in LaTeX
Reuse formulas/images/code/sources
Consistent style & references
Version control
Beamer [2] Template [3]

1. Re-use tooling, editor, setup (even this on overleaf alone much better that shared powerpoint presentation)
2. This presentation, report and thesis all live in one git-repository and share resources
3. Everything I ever work on is hosted in Git.
   - Without Version Control it is no longer possible for me to work effectively
   - Every workflow is massively ensured by it
   - Also for single-person work, but built in collaboration, synchronization and backup

# Beamer: LaTeX way of writing Presentations

## Presentations? - Custom Beamer Template

- Write presentations like your papers/thesis in LaTeX
- Reuse formulas/images/code/sources
- Consistent style & references
- Version control
- Easier collaboration

Beamer [2] Template [3]





Beamer: LaTeX way of writing Presentations
Presentations? - Custom Beamer Template
- Write presentations like your papers/thesis in LaTeX
- Reuse formulas/images/code/sources
- Consistent style & references
- Version control
- Easier collaboration
Beamer [2] Template [3]

1. Re-use tooling, editor, setup (even this on overleaf alone much better that shared powerpoint presentation)
2. This presentation, report and thesis all live in one git-repository and share resources
3. Everything I ever work on is hosted in Git.
   - Without Version Control it is no longer possible for me to work effectively
   - Every workflow is massively ensured by it
   - Also for single-person work, but built in collaboration, synchronization and backup

# Minimal example for beamer presentation

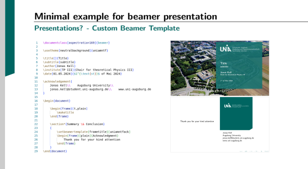
## Presentations? - Custom Beamer Template

```
\documentclass[aspectratio=169]{beamer}

\usetheme[neutralbackground]{uniamntf}

\title[]{Title}
\subtitle{subtitle}
\author{Jonas Kell}
\institute[TP III]{Chair for theoretical Physics III}
\date[01.05.2024]{$1^{\text{st}}$ of Mai 2024}

\acknowledgement{
    Jonas Kell\\     Augsburg University\\
    jonas.kell@student.uni-augsburg.de\\     www.uni-augsburg.de
}

\begin{document}

    \begin{frame}[t,plain]
        \maketitle
    \end{frame}

    \section*{Summary \& Conclusion}
    {
        \setbeamertemplate{frametitle}[uniamntfack]
        \begin{frame}[plain]{Acknowledgment}
            Thank you for your kind attention
        \end{frame}
    }
\end{document}
```









1. 29 lines gets you a basic presentation
2. First time very much slower as putting together by hand in PowerPoint
3. If you want to do crazy stuff, possible but really hard
4. Actually quite a timesaver when you have an example/template to work of and do not require crazy levels of features
5. When it works, perfectly portable (just a pdf), high performant and versatile (output rendering of presentation, notes, animations from one source)
6. If proper presentation tools, has all the bells and whistles (presentation timer, pointer, multi-view presentation and more)

Thank you for your kind attention

Universität Augsburg
Mathematisch-Naturwissenschaftlich-Technische Fakultät

Jonas Kell
University of Augsburg
jonas.kell@student.uni-augsburg.de
www.uni-augsburg.de



1. Thank you for your kind attention
2. All tools and other resources are referenced in the presentation
3. You can also find everything on my Github

# References I

## Summary & Conclusion


[1] J. Kell, *Math-manipulator tool - info and repository*, (2024) `https://github.com/jonas-kell/math-manipulator`.

[2] *CTAN: package beamer*, `https://ctan.org/pkg/beamer` (visited on 04/11/2024).

[3] J. Kell, *Https://github.com/jonas-kell/beamer-theme-unia-mntf*, (2024) `https://github.com/jonas-kell/math-manipulator`.

[4] J. Kell, *Latex sources for this presentation, the thesis and report*, (2024) `https://github.com/jonas-kell/master-thesis-documents`.

[5] J. Kell, *Thesis code implementation and reference*, (2024) `https://github.com/jonas-kell/master-thesis-code`.

[6] J. Kell, *Supplementary calculations perfomed with math-manipulator*, (2024) `https://github.com/jonas-kell/master-thesis-mm-calculations`.

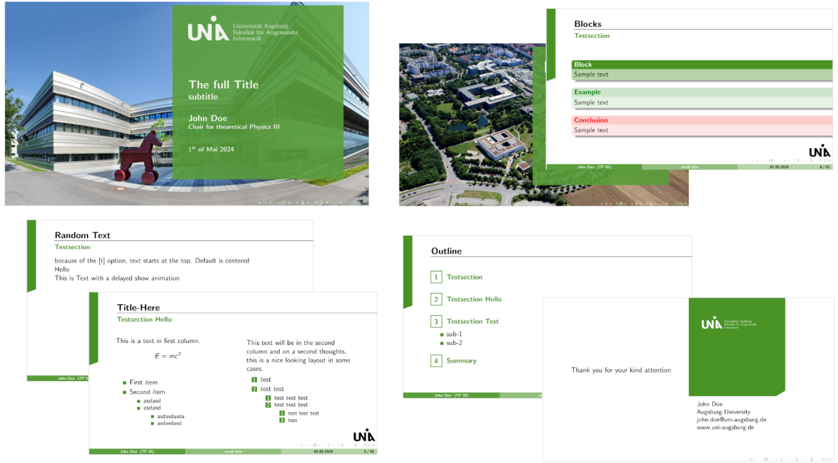



2025-02-11

Computer aided calculations
└Summary & Conclusion
└References

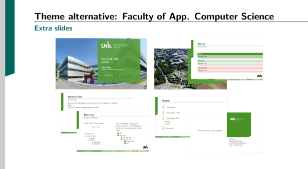

References I
Summary & Conclusion
[1] J. Kell, Math-manipulator tool - info and repository, (2024) https://github.com/jonas-kell/math-manipulator.
[2] CTAN: package beamer, https://ctan.org/pkg/beamer (visited on 04/11/2024).
[3] J. Kell, Https://github.com/jonas-kell/beamer-theme-unia-mntf, (2024) https://github.com/jonas-kell/math-manipulator.
[4] J. Kell, Latex sources for this presentation, the thesis and report, (2024) https://github.com/jonas-kell/master-thesis-documents.
[5] J. Kell, Thesis code implementation and reference, (2024) https://github.com/jonas-kell/master-thesis-code.
[6] J. Kell, Supplementary calculations perfomed with math-manipulator, (2024) https://github.com/jonas-kell/master-thesis-mm-calculations.

## Extra slides





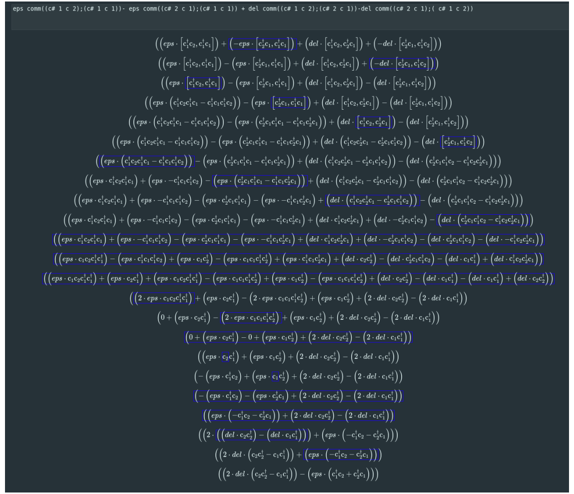


1. Template does also do "Faculty of Applied Computer Science" presentations out of the box
2. Alternate styles (e.g. title slide) available

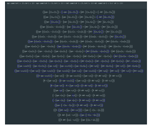
# Backup-Solution Problem 1

## Extra slides



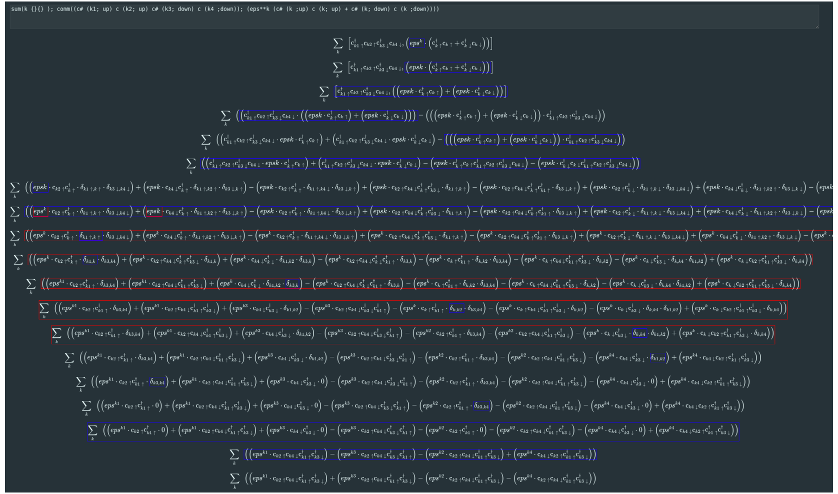




# Backup-Solution Problem 2

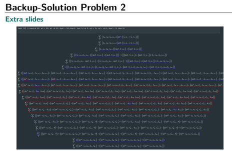
## Extra slides





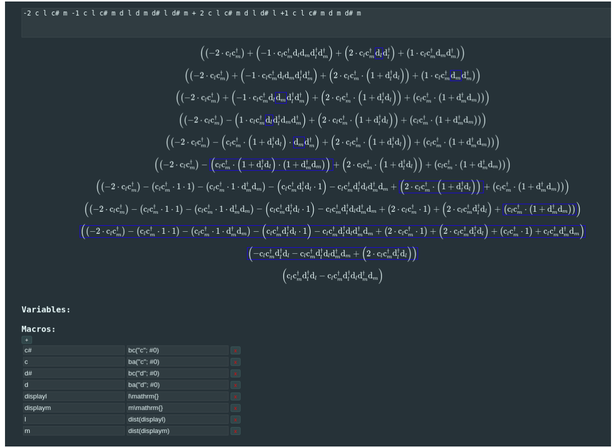



└Backup-Solution Problem 2



# Backup: Simplification V-Parts

## Extra slides



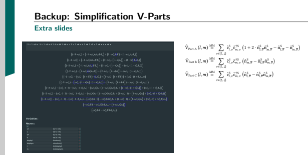

$$\hat{\mathrm{V}}_{\mathrm{Part\ A}}(l,m) \overset{\mathrm{MM}}{=} \sum_{\sigma\in\{\uparrow,\downarrow\}} \hat{\mathrm{c}}^{\mathrm{S}}_{l,\sigma}\hat{\mathrm{c}}^{\dagger\mathrm{S}}_{m,\sigma}\left(1 + 2\cdot\hat{\mathrm{n}}^{\mathrm{S}}_{l,\bar{\sigma}}\hat{\mathrm{n}}^{\mathrm{S}}_{m,\bar{\sigma}} - \hat{\mathrm{n}}^{\mathrm{S}}_{l,\bar{\sigma}} - \hat{\mathrm{n}}^{\mathrm{S}}_{m,\bar{\sigma}}\right)$$

$$\hat{\mathrm{V}}_{\mathrm{Part\ B}}(l,m) \overset{\mathrm{MM}}{=} \sum_{\sigma\in\{\uparrow,\downarrow\}} \hat{\mathrm{c}}^{\mathrm{S}}_{l,\sigma}\hat{\mathrm{c}}^{\dagger\mathrm{S}}_{m,\sigma}\left(\hat{\mathrm{n}}^{\mathrm{S}}_{m,\bar{\sigma}} - \hat{\mathrm{n}}^{\mathrm{S}}_{l,\bar{\sigma}}\hat{\mathrm{n}}^{\mathrm{S}}_{m,\bar{\sigma}}\right)$$

$$\hat{\mathrm{V}}_{\mathrm{Part\ C}}(l,m) \overset{\mathrm{MM}}{=} \sum_{\sigma\in\{\uparrow,\downarrow\}} \hat{\mathrm{c}}^{\mathrm{S}}_{l,\sigma}\hat{\mathrm{c}}^{\dagger\mathrm{S}}_{m,\sigma}\left(\hat{\mathrm{n}}^{\mathrm{S}}_{l,\bar{\sigma}} - \hat{\mathrm{n}}^{\mathrm{S}}_{l,\bar{\sigma}}\hat{\mathrm{n}}^{\mathrm{S}}_{m,\bar{\sigma}}\right)$$







1. Better optimization was found, rewriting in number operators DOES save on terms (however only for B and C, A still same number of terms and computational requirement, even if it looks nicer)
2. Doesn't help at all with computing more efficiently, so python script still exceptionally usefull
3. Wouldn't have come so far, if simple "brute-force" solution wasn't done in the first place