

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

## └ Outline

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

# Computer aided calculations

## └ Introduction of the problem

### └ Physics of what we calculate

- Content of my Practical Training (Fachpraktikum)

- Hubbard Model Hamiltonian

- System under influence of external electric field

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

$$\mathcal{H}_0 = U \cdot \sum_i \hat{a}_{i,\uparrow} \hat{a}_{i,\downarrow} + \sum_{i,s} \left( \frac{\vec{E} \cdot \vec{n}}{n_i} \right) \hat{a}_{i,s}$$

$$\hat{V} = -J \cdot \sum_{\langle i,j \rangle, s} (\hat{c}_{i,s}^\dagger c_{j,s} + \hat{c}_{j,s}^\dagger c_{i,s})$$

1. Hubbard-Model Hamiltonian with externally applied electric field
2. Hard-Core Bosonic 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

# Computer aided calculations

## └ Introduction of the problem

### └ Physics of what we calculate

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



# Computer aided calculations

## └ Introduction of the problem

### └ Physics of what we calculate

#### Physics of what we calculate

##### Introduction of the problem

$$\begin{aligned} [\hat{c}_{i,\sigma}, \hat{c}_{j,\sigma'}] &= [\hat{c}_{i,\sigma}^\dagger, \hat{c}_{j,\sigma'}^\dagger] = 0 \\ [\hat{c}_{i,\sigma}, \hat{c}_{j,\sigma'}^\dagger] &= \delta(i,j) \cdot \delta(\sigma, \sigma') \end{aligned}$$

##### ■ Hard-Core Bosonic operators

■ (Would work analogously with Fermionic operators)

$$\hat{c}_{i,\sigma}^{(0)} \leftrightarrow \hat{c}_{i,\sigma}^{(0)}$$

$$\hat{c}_{i,\sigma}^{(0)} \leftrightarrow \hat{c}_{i,\sigma}^{(0)}$$

##### ■ Two spin-degrees of freedom rewritten in alternate notation

$$H_0 = U \cdot \sum_i \hat{c}_{i,\sigma}^\dagger \hat{c}_{i,\sigma} + \sum_i \omega_i \hat{a}_i^\dagger \hat{a}_i + \sum_i \epsilon_i \hat{b}_i^\dagger \hat{b}_i$$

$$\hat{V} = -J \cdot \sum_{\langle i,j \rangle} (\hat{c}_{i,\sigma}^\dagger \hat{c}_{j,\sigma} + \hat{c}_{i,\sigma}^\dagger \hat{c}_{j,\sigma}^\dagger + \hat{c}_{i,\sigma}^\dagger \hat{c}_{j,\sigma}^\dagger + \hat{c}_{i,\sigma}^\dagger \hat{c}_{j,\sigma}^\dagger)$$

## 1. Operators are hard-core bosonic

- standard commutation relations
- occupations are only either zero or 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

# Computer aided calculations

## └ Introduction of the problem

### └ Working on paper: A Computer-Scientists view

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/disadvantages 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 repetitive, trivial and unfunny to do, paper cannot automatize
- 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

# Computer aided calculations

## └ Introduction of the problem

### └ Examples from "Theoretical Solid State Physics"

#### Examples from "Theoretical Solid State Physics"

##### Introduction of the problem

The image shows a handwritten derivation from a physics textbook. It starts with the expression  $\{S, \delta\} = \{c_1^{\dagger} c_2 - c_1^{\dagger} c_3, \delta (c_1^{\dagger} c_4 + c_2^{\dagger} c_3)\}$ . The derivation involves several steps of algebraic manipulation, including the use of commutation relations and simplification of terms. The final result is given as  $\frac{3}{2} V + 2J (c_1^{\dagger} c_2 - c_1^{\dagger} c_3)$ .

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

## Computer aided calculations

### └ Introduction of the problem

### └ Examples from "Theoretical Solid State Physics"

#### Examples from "Theoretical Solid State Physics"

##### Introduction of the problem



1. From the lecture "Theoretical Solid State Physics" (Theoretische Festkörperphysik), held 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)

# Computer aided calculations

## └ Introduction of the problem

### └ Examples from the Practical Training

#### 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

$$\hat{V} = -J \cdot \sum_{\mu} \left( \hat{c}_\mu^\dagger \hat{c}_\mu + \hat{c}_\mu^\dagger \hat{c}_\nu + \hat{c}_\nu^\dagger \hat{c}_\mu + \hat{c}_\nu^\dagger \hat{c}_\nu \right)$$

$$\hat{d}_{\mu}^{in}(t) = e^{i\omega_{\mu} t} \left( 1 + (e^{i\omega_{\mu} t} - 1) \hat{d}_{\mu}^{in} \hat{d}_{\mu}^{in\dagger} \right) \hat{d}_{\mu}^{in}$$

$$\hat{d}_{\mu}^{in}(0) = e^{i\omega_{\mu} t} \left( 1 + (e^{i\omega_{\mu} t} - 1) \hat{d}_{\mu}^{in} \hat{d}_{\mu}^{in\dagger} \right) \hat{d}_{\mu}^{in}$$

$$\hat{d}_{\mu}^{in}(t) = e^{i\omega_{\mu} t} \left( 1 + (e^{i\omega_{\mu} t} - 1) \hat{d}_{\mu}^{in} \hat{d}_{\mu}^{in\dagger} \right) \hat{d}_{\mu}^{in}$$

$$\hat{d}_{\mu}^{in}(0) = e^{i\omega_{\mu} t} \left( 1 + (e^{i\omega_{\mu} t} - 1) \hat{d}_{\mu}^{in} \hat{d}_{\mu}^{in\dagger} \right) \hat{d}_{\mu}^{in}$$

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  $\{\cdot\}(t) = e^{iH_0 t} \{\cdot\} e^{-iH_0 t}$

## Computer aided calculations

### └ Solutions: Off the shelf

#### └ Paper-like writing on digital devices

##### Paper-like writing on digital devices

###### Solutions: Off the shelf

■ Can give benefits that paper lacks

- Copy/Paste
- Collaborate
- Convert to text
- Saving and ordering large amounts of notes

■ HOWEVER

- Still no proper automatization
- Early access is lost when using proprietary technology
- "Vendor-Locking"

###### TIP:

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

1. Taking notes digitally 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 aided calculations

### └ Solutions: Off the shelf

#### └ Computer-Algebra Systems

## 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, reoccurring 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 aided calculations

### └ Math-Manipulator

#### └ What did I do?

##### └ What is Math-Manipulator?

### What is Math-Manipulator?

Math-Manipulator - What did I do?

The screenshot shows a web-based interface for a computer algebra system. At the top, there's a navigation bar with links like "Get Form", "Get File", "Get Previous", "Get Next", "Help", "Replace all", "Replace with Export", "Replace Variable", "Compute with Substitution", "Find Numbers", "Using Tools", "File", "Edit", "View", and "Help". Below the navigation, there's a text input field containing a mathematical expression:  $\sum_{k=1}^n \int_a^b \left( 123 + \frac{456}{x+789} \right) dx$ . To the right of the input field is a large button labeled "Compute". Below the input field, there's a preview area showing the derivative of the expression:  $\sum_{k=1}^n \int_a^b \left( 123 + \frac{456}{x+789} \right)^2 dx$ . At the bottom of the page, there's a footer with the text "Math-Manipulator [self@MathManipulator]".

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
  - → 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

## Computer aided calculations

### └ Math-Manipulator

#### └ How can you use it

##### └ In the 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

## Computer aided calculations

### └ Math-Manipulator

#### └ How can you use it

##### └ Usage as VS-Code Extension

### 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

## Computer aided calculations

### └ Math-Manipulator

#### └ Interactive Demonstration

##### └ Demo Problems from before

## Demo Problems from before

Math-Manipulator - Interactive Demonstration

$$\begin{aligned} & \cancel{\epsilon \{ c_0 t_{c_1} c_1^t c_2 \}} - \cancel{\{ c_2 t_{c_1} c_1^t c_0 \}} + \cancel{\delta \{ c_1^t t_{c_2} c_2^t c_0 \}} - \cancel{\delta \{ c_1^t c_1 c_2^t c_0 \}} \\ & = \cancel{\epsilon \{ c_0 t_{c_2} c_2^t c_1 \}} + 2\delta \{ c_0^t c_1 c_2^t c_1 \} \\ & \quad - (\cancel{c_0 \cdot t_{c_1} c_1^t c_2} - \cancel{c_0 \cdot t_{c_2} c_2^t c_1}) \cdot c_0^t c_1 c_2^t c_1 \end{aligned}$$

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)
```

# Computer aided calculations

- Math-Manipulator

- Interactive Demonstration

- Usage for the Practical Training

## Usage for the Practical Training

### Math-Manipulator - Interactive Demonstration

$$\begin{aligned}
 V(t) &= \left\{ V^i(t) \right\} i = 1, \dots, \sum_{k=1}^n \left( \left\{ \tilde{V}_k^{i_1}, \dots, \tilde{V}_k^{i_m} \right\} \right) \partial_i \\
 &= -J \cdot \sum_{k=1}^n \left( \tilde{V}_k^1 \tilde{V}_k^2 \tilde{V}_k^3(t) + \tilde{V}_k^3 \tilde{V}_k^1(t) \right) \\
 &\stackrel{\text{def}}{=} -J \cdot \sum_{k=1}^n \left[ e^{i \omega_k t - \alpha_k}, V_{\text{max}}(t, \omega_k) + e^{i \omega_k t - \alpha_k}, V_{\text{min}}(t, \omega_k) + \right. \\
 &\quad \left. e^{i \omega_k t - \alpha_k}, V_{\text{peak}}(t, \omega_k) + e^{i \omega_k t - \alpha_k}, V_{\text{trough}}(t, \omega_k) \right] \\
 V_{\text{max}}(t, \omega_k) &\stackrel{\text{def}}{=} \left( 1 - \tilde{V}_k^1(t) \right)^2 + \left( 1 - \tilde{V}_k^2(t) \right)^2 + \left( 1 - \tilde{V}_k^3(t) \right)^2 + \\
 &\quad + \left( -1 - \tilde{V}_k^1(t) \right)^2 + \left( -1 - \tilde{V}_k^2(t) \right)^2 + \left( -1 - \tilde{V}_k^3(t) \right)^2 + \left( -1 - \tilde{V}_k^1(t) \right)^2 + \\
 V_{\text{min}}(t, \omega_k) &\stackrel{\text{def}}{=} \left( -1 - \tilde{V}_k^1(t) \right)^2 + \left( -1 - \tilde{V}_k^2(t) \right)^2 + \left( -1 - \tilde{V}_k^3(t) \right)^2 + \left( -1 - \tilde{V}_k^1(t) \right)^2 + \\
 &\quad + \left( 1 - \tilde{V}_k^1(t) \right)^2 + \left( 2 - \tilde{V}_k^2(t) \right)^2 + \left( 2 - \tilde{V}_k^3(t) \right)^2 + \left( 2 - \tilde{V}_k^1(t) \right)^2 + \\
 V_{\text{peak}}(t, \omega_k) &\stackrel{\text{def}}{=} \left( -2 - \tilde{V}_k^1(t) \right)^2 + \left( -2 - \tilde{V}_k^2(t) \right)^2 + \left( -2 - \tilde{V}_k^3(t) \right)^2 + \left( -2 - \tilde{V}_k^1(t) \right)^2 + \\
 &\quad + \left( 2 - \tilde{V}_k^1(t) \right)^2 + \left( 1 - \tilde{V}_k^2(t) \right)^2 + \left( 1 - \tilde{V}_k^3(t) \right)^2 + \left( 1 - \tilde{V}_k^1(t) \right)^2
 \end{aligned}$$

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 upfront configuration (and in my case programming) effort, re-calculating the endresult took 20min instead of one week

# Computer aided calculations

- Custom Python Scripts (SymPy)

- Optimization: Difference of  $V(t)$  for Hopping

## Optimization: Difference of $V(t)$ for Hopping

### Custom Python Scripts (SymPy)

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

- These require differences of  $V^1(t)$  and  $E_0$

- Too many terms for simple evaluation by hand

- Intelligent pre-computation required to speed up processing

$$\begin{aligned}
 a &= \frac{p(S, t)}{p(N, t)} = \frac{f(S, t)}{f(N, t)} \cdot \frac{\sum_{k \in S} n_{\text{act}(k, t)}^2 / p_k}{\sum_{k \in N} n_{\text{act}(k, t)}^2 / p_k} \\
 &= \frac{\left| p_k \right|^2 \cdot n_{\text{act}(k, t)}^2 / p_k}{\left| p_k \right|^2 \cdot n_{\text{act}(k, t)}^2 / p_k + \sum_{k' \neq k} n_{\text{act}(k', t)}^2 / p_{k'}} \\
 &= \frac{\left| p_k \right|^2 \cdot n_{\text{act}(k, t)}^2 / p_k}{\left| p_k \right|^2 \cdot n_{\text{act}(k, t)}^2 / p_k + n_{\text{act}(k', t)}^2 / p_{k'}} \\
 &= \frac{\left| p_k \right|^2 \cdot n_{\text{act}(k, t)}^2 / p_k}{\left| p_k \right|^2 \cdot n_{\text{act}(k, t)}^2 / p_k + n_{\text{act}(k, t)}^2 / p_k} \\
 &= \frac{\left| p_k \right|^2 \cdot n_{\text{act}(k, t)}^2 / p_k}{\left| p_k \right|^2} \\
 &= \frac{\left| p_k \right|^2 \cdot n_{\text{act}(k, t)}^2}{\left| p_k \right|^2} \\
 &= \frac{\left| p_k \right|^2 \cdot (n_{\text{act}(k, t)} - n_{\text{act}(k, t)})}{\left| p_k \right|^2} \\
 &= 0
 \end{aligned}$$

# Computer aided calculations

- Custom Python Scripts (SymPy)

- The process: Difference of  $V(t)$  for Hopping

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

### Custom Python Scripts (SymPy)

$$E(N) - E(\bar{N}) = U \sum_{i,j} \langle r_i, r_j \rangle \cdot d_i \cdot d_j \cdot \sum_{\sigma_i, \sigma_j} \langle \sigma_i, \sigma_j \rangle \cdot \sum_{k,l} \langle r_k, r_l \rangle$$

$$\stackrel{\text{def}}{=} (d - \bar{d}) \left( \langle v_{1,0}, -v_{1,0} \rangle + \dots + \begin{cases} \langle v_{1,0}, -v_{1,1} \rangle \langle v_{1,1}, -v_{1,0} \rangle & : v_1 = v_0 \\ \langle v_{1,1}, -v_{1,0} \rangle \langle v_{1,0}, -v_{1,1} \rangle & : v_1 \neq v_0 \end{cases} \right) \cdot d_{1,0}$$

$$= (d - \bar{d}) \langle v_{1,0}, -v_{1,0} \rangle + U \langle v_{1,0}, -v_{1,0} \rangle \langle v_{1,0}, -v_{1,0} \rangle$$

Setup difference

Replace one by appropriate new occupation

Simplify as much as possible

Sum over all possible neighbor combinations

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

- The  $V$  differences are  $3 \cdot 8 \cdot 32 \cdot 4 = 96 \cdot 2 = 192$  terms
- The process is relatively straight forward:
  - write down the terms
  - replace
  - simplify difference
- Still quite complicated and a lot of work, but:
  - way faster to write because non-repetitive 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

## Computer aided calculations

### └ Custom Python Scripts (SymPy)

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

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

Custom Python Scripts (SymPy)

Generating Script  
`simplificationtermhelper.py`

Generated Script  
`analyticalcalcfuctions.py`

1. `simplificationtermhelper.py` script to generate the script
2. `analyticalcalcfuctions.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

## Computer aided calculations

### └ Presentations? - Custom Beamer Template

#### └ Beamer: LaTeX way of writing Presentations

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

1. Re-use tooling, editor, setup (even this on overleaf alone much better than 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

## Computer aided calculations

### └ Presentations? - Custom Beamer Template

#### └ Minimal example for beamer presentation

#### Minimal example for beamer presentation

##### Presentations? - Custom Beamer Template



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)

# Computer aided calculations

## └ Summary & Conclusion

### └ Acknowledgment

Thank you for your kind attention



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

## Computer aided calculations

### └ Extra slides

#### └ Theme alternative: Faculty of App. Computer Science

Theme alternative: Faculty of App. Computer Science

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

## Computer aided calculations

### └ Extra slides

#### └ Backup: Simplification V-Parts

## Backup: Simplification V-Parts

### Extra slides



$$\hat{v}_{max}(l,m) = \sum_{n=0}^{\infty} \hat{v}_n^l \hat{v}_{m,n}^l (1 + z^l \hat{v}_n^l \hat{v}_{m,n}^l - \hat{v}_{m,n}^l)$$
$$\hat{v}_{max}(l,m) = \sum_{n=0}^{\infty} \hat{v}_n^l \hat{v}_{m,n}^l (\hat{v}_{m,n}^l - z^l \hat{v}_n^l \hat{v}_{m,n}^l)$$
$$\hat{v}_{max}(l,m) = \sum_{n=0}^{\infty} \hat{v}_n^l \hat{v}_{m,n}^l (1 + z^l \hat{v}_n^l \hat{v}_{m,n}^l - \hat{v}_{m,n}^l)$$

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 useful
3. Wouldn't have come so far, if simple "brute-force" solution wasn't done in the first place