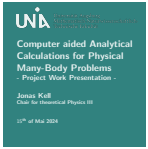
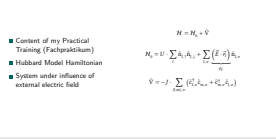
Computer aided calculations



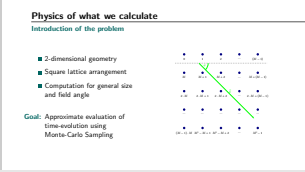
1. Welcome
2. Presentation of "Project Work" (Projektarbeit)
3. Runs parallel to the "Practical Training" (Fachpraktikum)
4. Presentation in this Group-Slot

Outline

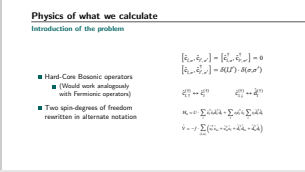
1. Introduction of theme worked on in Project-Work, Practical Training and Master Thesis
   - The mathematical Problem of Many-Body Physics
2. Focus of the report is on the mathematical details. I have a history in Computer-Science, therefore would like here to present some techniques I learned in my 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



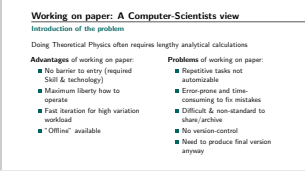
1. Hubbard-Model Hamiltonian with exterally applied electric field
2. Hard-Core Bosinic Operators (will be explained in later slide, do not need to elaborate on)
3. Spin-dependent lattice Model
4. External electric field acts symmetric on both spin directions
   - This will be important later, multiple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



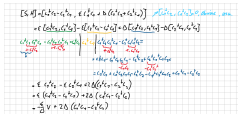
1. 2-dimensional square geometry
2. Program supports already for general size inputs
3. Thanks to the Monte-Carlo Sampling we can even efficiently evaluate larger systems already



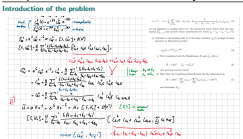
1. Operators are hard-core bosonic
   - standard commutation relations
   - occupations are only either zero ore one
2. fermionic operators (anti-commutation) would work equivalent
3. other notation to rewrite the two spin degrees of freedom into two operators of different dof makes it more readable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



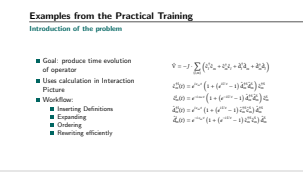
1. Comparison of advantages/disatvantages of doing calculations "by hand"
2. Advantages
   - No skills required to learn, writing is natural, every new symbol can just be written
   - Operational Libery (cross out stuff as please, swap whatever without "proof")
   - For completely different approaches 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



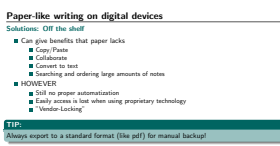
1. From the lecture "Theoretical Solid State Physics" (Theoretische Festkörperphysik), helt by Markus Heyl in WS 22/23
2. Sheet 3, Problem 2: Schrieffer-Wolf transformation
3. Rather simple calculation, still contains sing mistakes (that here luckily averaged out)



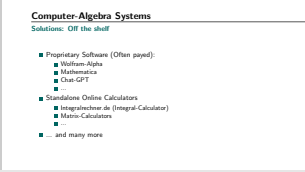
1. From the lecture "Theoretical Solid State Physics" (Theoretische Festkörperphysik), helt by Markus Heyl in WS 22/23
2. Sheet 8, Problem 1: Schrieffer-Wolf transformation of the Hubbard Model
3. Just written down steps done by someone else (Tutor/other pupil, often shared the terms to calculate them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



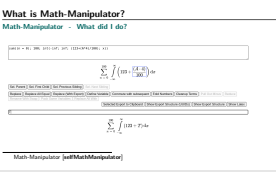
1. Want to calculate the Time-Evolution of the operator V in the interaction Picture
2. Basically comes down to inserting definitions (however already derived from MM)
3. Distributing and re-ordering is a painful, time-intensive process
4. How to time-evolve operator in the Interaction Picture $\{\cdot\}(t)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



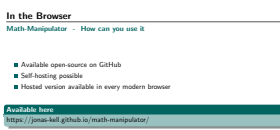
1. Taking notes digially can have many benefits
2. OneNote, Xournalpp, Notion and many more examples
3. Still lacks "true" automatization
4. Great danger of Vendor-Locking, losing access
   - Cloud-Driven offers do not work without internet
   - Often times even tied to the device (Apple-Notes I-pad, 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.



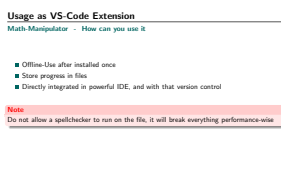
1. There exist already countless possible attempts at solving the worlds problems
2. All of them are great at what they are designed for, I will not attempt competing with any of the m in terms of any metric except maybe usability (and probably still fail)
3. Chat-GPT is included in the list, because 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



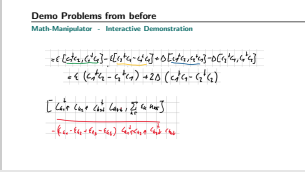
1. I wrote a custom tool that should help with the process of doing math paper-like, but with the help of computers
2. Features/Properties
   - Relatively smart initial text-input
     - Omit implied multiplication
     - Omit implied zeros
     - Parameter autoboxing
     - Bracket culling
     - → 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



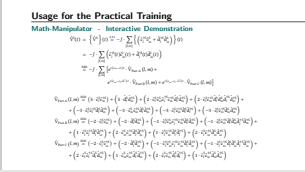
1. Source code available
2. (Relatively) easily extensible, tutorial and issues available
3. Instantly usable, offline and online



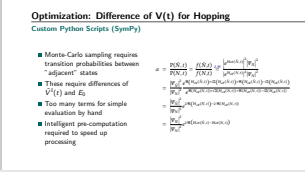
1. For more serious use, integrated in the code-editor VS-Code as an Extension
2. Allows for all features, but includes
   - Offline use out of the box
   - Storage of files (with that version control)
   - Multiple parallel projects
3. Has already taken entry into my personal workflow by now
   - Probably 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



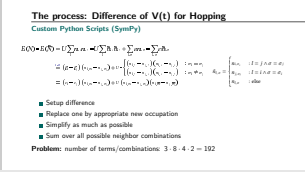
1. Go to Demo in VS-Code folder math-manipulator-calculations in the presentation subfolder
2. Think of disabling spellchecker
3. Think to first split deltas before evaluating !!!!!!
4. Macros and other optimizations are possible, but not really necessary
5. Input for simple demo: distribute, cleanup, 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)`



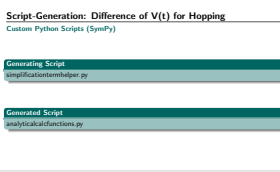
1. Go to Demo in VS-Code submodule mm-calculations.
2. Think of disabling spellchecker
3. Show process of how V is calculated
4. Story: at the end at first result was no longer symmetrical in up and down, but it should have been. Mistake was in the very first input. But because of the upfornt configuration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



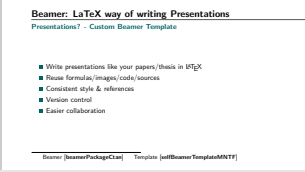
1. "Adjacent" states because from one Monte-Carlo step to the next we require a small change of one or two particle-sites. Could come from hopping or flipping.
2. The E_0 differences are quickly calculated by hand in the report



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



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 $\rightarrow$ simplification with Sympy
5. Parallel to computer-science practice "Vendored" code



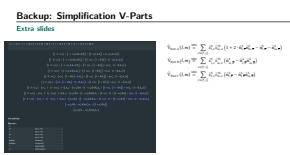
1. Re-use tooling, editor, setup (even this on overleaf alone much better that shared powerpoint presentation)
2. This presentation, report and thesis all live in one git-repository and share resources
3. Everything I ever work on is hosted in Git.
   - Without Version Control it is no longer possible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

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

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

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



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