Vysoká škola chemicko-technologická, Praha
Fakulta chemického inženýrství
Ústav fyzikální chemie (403)

Isingův model ve 3D

P02-ISING

Jakub Vencel
Semestrální práce
Počítačová chemie (B403011)

Praha 2026
vedoucí práce: prof. RNDr. Jiří Kolafa, CSc.

Obsah

1 Zadání

Dostupné z https://github.com/mhkoscience/pchem/blob/main/projekty/P02-ISING/README.yaml.

2 Úvod

2.1 Isingův model

Předpokládejme kubickou mřížku $\mathbb{Z}^3$ o velikosti $L \times L \times L$. V takové mřížce se nachází $N = L^3$ prvků, které mají spin $\sigma = \{-1; +1\}$ [1]. Pro tuto mřížku můžeme definovat Hamiltonián $H(\sigma)$ (platí pro případ, kdy je nulové vnější magnetické pole)

$$H(\sigma) = -J \sum_{<ij>} (\sigma_i \sigma_j), \tag{1}$$

kde J je interakční energie, která nabývá hodnot $J > 0$ pro feromagnety a $J < 0$ pro antiferomagnety a sčítá se přes nejbližší sousedy na mřížce.

Pro feromagnetické látky musí platit, že konfigurace spinů je taková, aby vznikl nenulový magnetický moment M, který lze vypočítat jako součet všech spinů v mřížce

$$M(\sigma) = \sum_{i=1}^{N} \sigma_i. \tag{2}$$

Obdobně lze vypočítat energii spinové konfigurace mřížky E [2]

$$e_i = -\frac{J}{2} \sum_{<ij>} (\sigma_i \sigma_j), \tag{3}$$

$$E(\sigma) = \sum_{i=1}^{N} e_i. \tag{4}$$

Spiny v mřížce mají uspořádání podle Boltzmannova rozdělení dané partiční funkcí

$$p(\boldsymbol{\sigma}|T) = \exp\left(-\frac{E(\sigma)}{\mathbf{k_B} T}\right), \tag{5}$$

kde $\mathbf{k_B}$ je Boltzmannova konstanta a T je teplota [3]. Do kritické teploty T_c (někdy zvané Curieova teplota) jsou spiny v dostatečném uspořádání, aby magnetický moment $M(\sigma)$ měl nenulovou hodnotu. Při teplotě T_c dochází k fázové přeměně druhého druhu a v teplotách $T > T_\mathrm{c}$ se moment ztrácí [4]. Pro 3D Isingův model byla inverzní kritická teplota numericky vypočtena s výsledkem $\beta_\mathrm{c} = 0.2216595 \pm 0.0000026$ [5].

2.2 Monte Carlo

Jelikož neexistuje přijímané analytické řešení Isingova modelu pro 3D mřížku, používá se k simulaci numerická metoda Monte Carlo (MC simulace).

2.2.1 Typy algoritmů

Metropolisův algoritmus je založen na generování náhodného čísla představující pravděpodobnost $p(\sigma_i|T)$ (přepsáno pro stavy a a b $p_{a\to b}$). Pokud platí $\Delta E(\sigma_i) < 0$, proveď změnu stavu $a \to b$ s pravděpodobností $p_{a\to b} = 1$. V opačném případě proveď změnu stavu, pokud je vygenerovaná pravděpodobnost menší než partiční funkce $p_{a\to b} < \exp(-E(\sigma)/\mathbf{k_B}T)$ [2].

Heat Bath algoritmus na rozdíl od Metropolisova algoritmu, který sestává ze tří částí (navrhnutí změny, vyhodnocení přijetí, změna stavu $a \to b$), rovnou přepisuje stav na základě Boltzmannova rozdělení podle pravděpodobností [6]

$$\Delta E(\sigma) = 2J \sum_{<ij>} (\sigma_i \sigma_j)\,, \tag{6}$$

$$P(\sigma = +1) = \frac{1}{1 + \exp\left(-\frac{\Delta E(\sigma)}{\mathbf{k_B}T}\right)}, \tag{7}$$

$$P(\sigma = -1) = \frac{1}{1 + \exp\left(\frac{\Delta E(\sigma)}{\mathbf{k_B}T}\right)}. \tag{8}$$

Jiné algoritmy, kterých je celá řada, například mění celé klastry spinů.

3 Program

Ačkoliv samotný MC algoritmus je jednoduchý, 3D prostor s sebou přináší vysoký počet iterací, které je potřeba napočítat: $i = 2 \cdot L^3 \cdot t \cdot eq$. Při použitých parametrech $L = 50$, počet iterací $t = 4096$, ekvilibrační kroky $eq = 10$ se dostáváme k cca deseti miliardám iteracím. Z tohoto důvodu jsem se rozhodl rozdělit program na dvě části.

Simulaci samotnou jsem proto implementoval do programovacího jazyka C a konfiguraci, export a vizualizaci výpočtu do Pythonu. Python při spuštění vytvoří několik binárních souborů, které přímo čte simulační program spuštěný jako subproces (eliminuje se nutnost konverze):

1. atoms.bin (8 bit integer) – 1D pole o délce L^3;
2. config.bin (struct) – L, t, J, inicializační metoda;
3. temp_cycle.bin (32 bit float) – teplotní cyklus.

```
# Linux
$ ./bin/simulant path_to_simulation
# Windows
  .\bin\simulant.exe path_to_simulation
```

3.1 Simulace

Byl napsán simulační program v programovacím jazyce C, do kterého byl implementován Heat Bath algoritmus. Metoda počítá s redukovanými veličinami: interakční energie $J = 1$, Boltzmannova konstanta $\mathbf{k_B} = 1$, redukovaná teplota $t = T/T_c$.

Aby bylo zaručeno, že se energie a magnetizace začnou počítat pro mřížku, jejíž konfigurace odpovídá Boltzmannovu rozdělení, byl přidán kromě možnosti náhodného rozdělení spinů i scénář WARM-UP. Na začátku jsou všechny spiny rovny -1. Pro tuto mřížku se provede počet iterací t, které odpovídají počtu iterací kýžené simulace. Teplotní cyklus WARM-UP začíná v nadkritické oblasti a pomalu se ochlazuje k teplotě, kterou začíná temp_cycle.bin [3].

Pro prevenci hystereze se v každé iteraci MC výpočet opakuje po definovaný počet ekvilibrčních kroků $eq = 10$ při stejné teplotě. Systém má tak čas přizpůsobit se nové teplotě.

3.2 Grafické uživatelské rozhraní

Grafické uživatelské rozhraní (GUI) bylo naprogramováno v Pythonu s využitím modulů NumPy (generování mřížky, teplotních cyklů a export do binárních souborů), customtkinter (grafické rozhraní) a matplotlib (vizualizace sledovaných vlastností a export animace).

Uživatel má možnost měnit typ a intenzitu teplotního cyklu:

1. konstantní teplota;
2. ohřev;
3. ochlazování;
4. pomalý ohřev a pomalé ochlazování;
5. rychlý ohřev a pomalé ochlazování;
6. pomalý ohřev a rychlé ochlazování;

dále hodnotu interakční energie, velikost mřížky a může si zvolit výše vysvětlenou WARM-UP metodu, nebo nechat NumPy vygenerovat náhodnou mřížku.

Grafické prvky použité v aplikaci byly buď převzaty z volně licencovaných příspěvků na IconScout[1], nebo dotvořeny v programu GIMP. Jako pozadí hlavního okna aplikace jsem zvolil Obrázek 3: Schematická reprezentace konfigurace 2D Isingova modelu na čtvercové mřížce[2].

4 Výsledky

Tabulka 1: Parametry simulací

Parametr	Hodnota
L	50
eq	10
J	1
t	4096
T_{min}	1.0
T_{max}	6.0
inicializace	WARM-UP

Není-li řečeno jinak, simulace jsem nastavoval podle tabulky 1.

[1] https://iconscout.com/

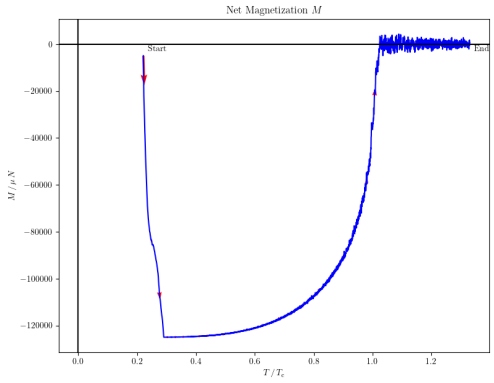
[2] Obrázek dostupný z https://www.researchgate.net/figure/Schematic-representation-of-a-configuration-of-the-2D-Ising-model-on-a-square-lattice_fig2_321920877

4.1 Konstantní teplota

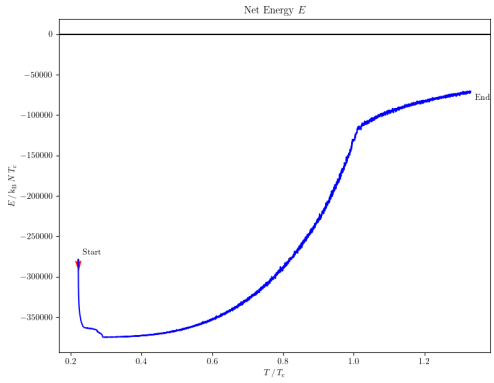
Při udržování konstantní teploty lze sledovat, že hodnota magnetizace i energie se v průběhu času v průměru nemění. Čím více se teplota blíží ke kritické teplotě, tím více se zvyšuje energie a klesá uspořádanost mřížky.

4.2 Zahřívání

Energie postupně stoupá, po kritické teplotě dochází ke zlomu. Magnetizace po překročení kritické teploty osciluje kolem nuly.



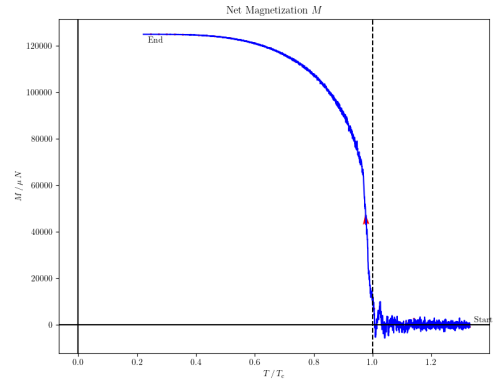
(a) Vývoj magnetického momentu v závislosti na teplotě



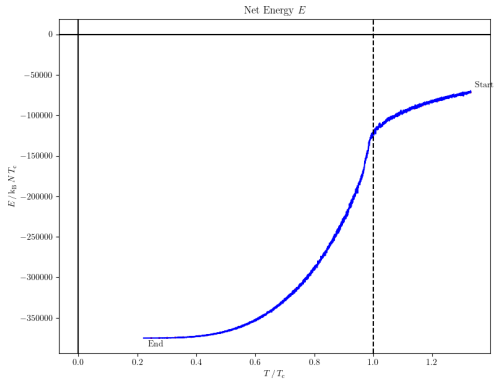
(b) Vývoj energie v závislosti na teplotě

4.3 Ochlazování

Během ochlazování postupně přechází neuspořádaný systém zpět do uspořádaného. Energie klesá a magnetický moment celé mřížky se náhodně ustálí v kladném, nebo záporném směru.

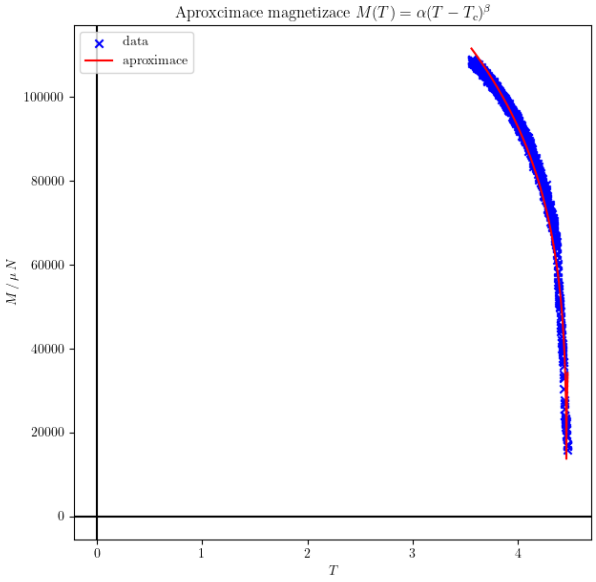


(a) Vývoj magnetického momentu v závislosti na teplotě



(b) Vývoj energie v závislosti na teplotě

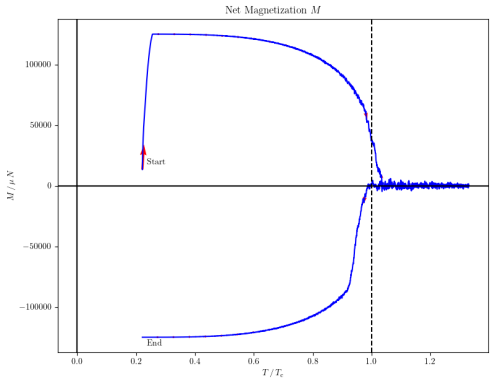
Data jsem proložil závislostí $M(T) = \alpha(T - T_c)^\beta$, ze které jsem odečetl kritickou teplotu $T_c = 4,462$.



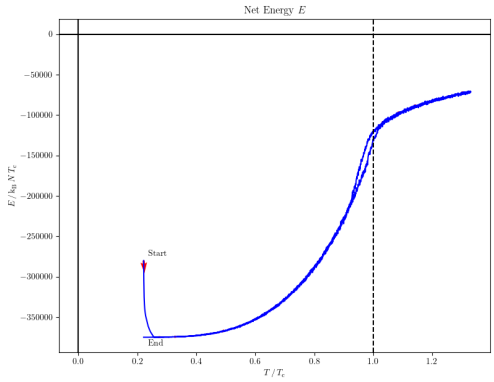
Obrázek 3: Získání kritických konstant

4.4 Tepelný cyklus

Tepelný cyklus byl nastaven na náhodnou počáteční konfiguraci.



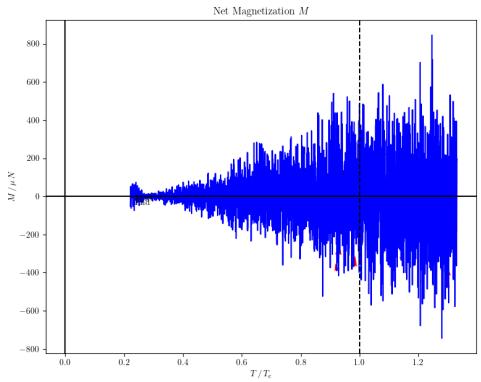
(a) Vývoj magnetického momentu v závislosti na teplotě



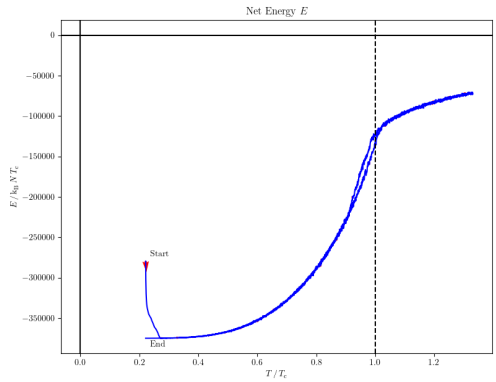
(b) Vývoj energie v závislosti na teplotě

4.5 Antiferomagnet

Pro simulaci antiferomagnetu jsem nastavil $J = -1$. Magnetizace po celou dobu simulace nevykazuje jednoznačný magnetický moment. V animaci je vidět, že žádné 2 spiny vedle sebe v ustáleném stavu nesměřují stejným směrem.



(a) Vývoj magnetického momentu v závislosti na teplotě

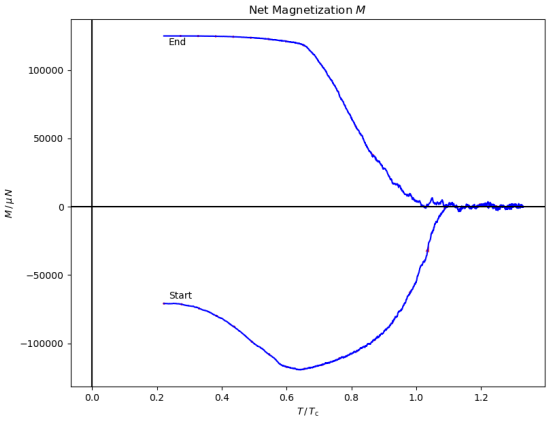


(b) Vývoj energie v závislosti na teplotě

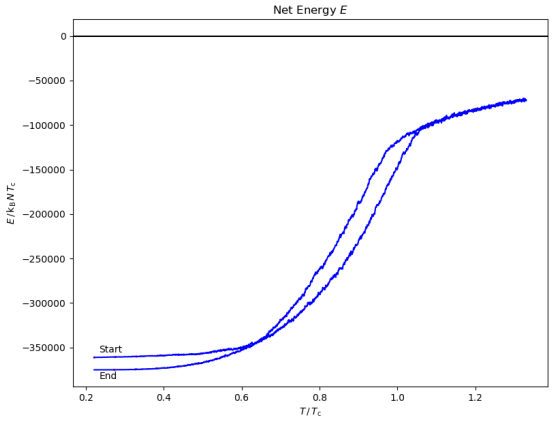
4.6 Hystereze – vliv počátečních podmínek na průběh simulace

Tepelný cyklus je závislý na počátečních podmínkách. Na začátku tohoto cyklu se v mřížce vyskytovalo několik metastabilních oblastí. Ty byly příčinou vzniku hysterezní křivky. Pro eliminaci tohoto jevu u dalších simulací jsem přidal pro každý iterační krok několik ekvilibračních mezikroků při stejné teplotě, aby se systém stihl nové teplotě přizpůsobit.[3]

[3] Hysterezi by pak také měl eliminovat WARM-UP, avšak s největší pravděpodobností z důvodu chyby paměťové segmentace při přechodu mezi WARM-UP a měřením dochází na začátku simulace k chybné konfiguraci.



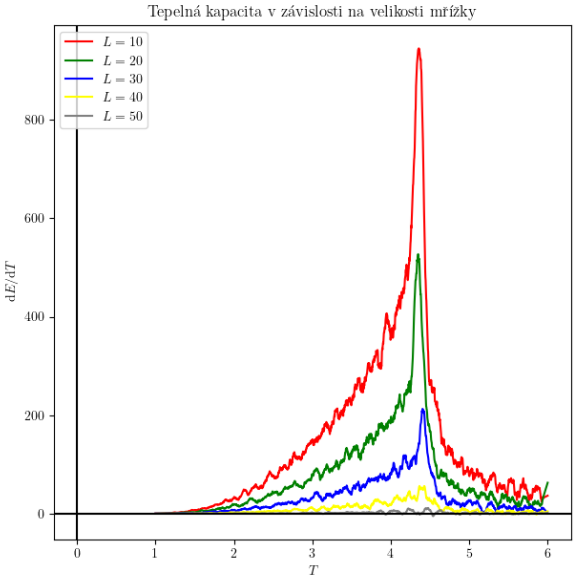
(a) Vývoj magnetického momentu v závislosti na teplotě



(b) Vývoj energie v závislosti na teplotě

4.7 Vliv velikosti mřížky na průběh simulace

Provedl jsem 5 simulací s velikostmi mřížky 10, 20, 30, 40 a 50. Velikost mřížky má vliv na přesnost odečítání kritické teploty. Pro znázornění jsem vypočetl tepelnou kapacitu $C = \frac{\mathrm{d}E}{\mathrm{d}T}$.



Obrázek 7: Tepelná kapacita

5 Závěr

Napsal jsem program pro MC simulaci Isingova feromagnetu na kubické mřížce v periodických okrajových podmínkách s nulovým vnějším magnetickým polem. Na této mřížce jsem monitoroval energii a celkovou magnetizaci. Simulaci jsem provedl pro různé termální cykly a různé velikosti mřížky.

Z poměru T/T_c jsem stanovil, že k fázovému přechodu došlo v $T/T_c \approx 1$, čili simulace správně odhadla $T_c \approx 4.5$.

Obrázek 8: Odkaz na GitHub úložiště (https://github.com/vencelj/MC_Ising)

Odkaz na zdrojové kódy a složkami s kompletními simulovanými daty i animacemi naleznete na odkazu výše.

Reference

1. VISWANATHAN, Gandhimohan M.; PORTILLO, Marco Aurelio G.; RAPOSO, Ernesto P.; LUZ, Marcos G. E. da. What Does It Take to Solve the 3D Ising Model? Minimal Necessary Conditions for a Valid Solution. *Entropy*. 2022, roč. 24, č. 11. ISSN 1099-4300. Dostupné z DOI: 10.3390/e24111665.
2. FITZPATRICK, Richard. *Computational Physics: The Ising Model* [https://farside.ph.utexas.edu/teaching/329/lectures/node105.html]. 2006. [cit. 2025-12-26].
3. UNIVERSITY OF CAMBRIDGE. *Computational Projects, Part II: 11.2 The Ising Model* [online]. 2024. [cit. 2025-12-26]. Tech. zpr. Faculty of Mathematics. Dostupné z: https://www.maths.cam.ac.uk/undergrad/catam/II/11pt2.pdf. Mathematical Tripos, Undergraduate Manual.
4. HASENBUSCH, M; PINN, K. , , , and from 3D Ising energy and specific heat. *Journal of Physics A: Mathematical and General*. 1998, roč. 31, č. 29, s. 6157. Dostupné z DOI: 10.1088/0305-4470/31/29/007.
5. FERRENBERG, Alan M.; LANDAU, D. P. Critical behavior of the three-dimensional Ising model: A high-resolution Monte Carlo study. *Phys. Rev. B*. 1991, roč. 44, s. 5081–5091. Dostupné z DOI: 10.1103/PhysRevB.44.5081.
6. LIPOWSKI, Adam; FERREIRA, António L.; LIPOWSKA, Dorota. Heat-Bath and Metropolis Dynamics in Ising-like Models on Directed Regular Random Graphs. *Entropy*. 2023, roč. 25, č. 12. ISSN 1099-4300. Dostupné z DOI: 10.3390/e25121615.

A Pseudokód MC

```
#include "monte_carlo.h"

float totalEnergy(struct SimConfig *sim, struct SimMeas *meas){
        energyTotal = 0;
        L = sim->L;
        J = sim->J;

        float energyLUT[7]; // look-up table for energy
        for (int s = -6; s <= 6; s += 2) {
                energyLUT[(s + 6) / 2] = -J/2.0f * (float)s;
        }

        for (all atoms in lattice){
                // periodic boundary conditions
                spin = meas->atoms[a*L*L + b*L + c];
                spinNeighbours = meas->atoms[(a-1)*L*L + j*L + k]
                + meas->atoms[(a+1)*L*L + j*L + k]
                + ...;

                spinConfiguration = spin * spinNeighbours;
                energyTotal += energyLUT[(spinConfiguration + 6) / 2];
        }
        return energyTotal;
}

void monteCarlo(struct SimConfig *sim, struct SimMeas *meas, iterCounter){
        L = sim->L;
        J = sim->J;
        N = (uint64_t)L*L*L;
        temp = meas->temps[iterCounter];
        energyIter;
        magnetizationIter;

        /*
        * if this is the first iteration ->
        * -> energyIter = totalEnergy()
        * if this is the first iteration after WARM-UP ->
        * -> energyIter = meas->energy[last]
        * else ->
        * -> energyIter = meas->energy[iterCounter - 1]
        * analogy for magnetizationIter
        */

        float energyLUT[7]; // look-up table for energy
        float probLUT[7]; // look-up table for probability
        for (int s = -6; s <= 6; s += 2) {
```

```
                float dE = 2.0f * J * (float)s;
                probLUT[(s + 6) / 2]=1/(1+exp(-dE/(bolzmannConst*temp)));
        }

        for (eq = 0; eq<10; eq++){ // equilibrium steps
                for (i=0; i<N;i++){
                        a, b, c = rand() % L;

                        // periodic boundary conditions
                        spin = meas->atoms[a*L*L + b*L + c];
                        spinNeighbours = meas->atoms[(a-1)*L*L + j*L + k]
                        + meas->atoms[(a+1)*L*L + j*L + k]
                        + ...;

                        if (PROB() < probLUT[(spinNeighbours + 6) / 2]) {
                                meas->atoms[a*L*L + b*L + c] = 1;
                        } else {
                                meas->atoms[a*L*L + b*L + c] = -1;
                        }

                        dE = -J * spinNeighbours * (newSpin - spin);
                        energyIter += dE;
                        magnetizationIter += (newSpin - spin);
                }
        }
        // save energyIter and magnetizationIter into struct meas
}

int main(int argc, char **argv){
        // load path to dir if argc == 2, else return 1;
        const char *dirPath = argv[1];
        // configure simulation -> load all binaries

        // call all monteCarlo() for i < iterations
        if (sim.init == WARM_UP){
                monteCarlo(&sim, &meas, i); // run MC at WARM-UP temp. cycle
                monteCarlo(&sim, &meas, i); // run simulation
        }
        else{
                monteCarlo(&sim, &meas, i); // run simulation
                // after every 10 iterations save state of atoms in atoms.bin
        }

        // create *.bin with calculated magnetization and energy
        // free allocated memory
        return 0; // end program
}
```