TP IV
Version 1 January 12, 2018

TP IV : Quantum critical scaling

Annina Riedhauser

Supervisor: Mingee Chung

Contents

Introduction

The study of quantum matter has become a great part of modern physics research. Quantum criticality appears in the vicinity of a quantum critical point where there is an interplay between quantum and thermal fluctuations. In the quantum critical region, 'exotic' phases therefore appear which might be the origin of unconventional superconductivity, for example. The study of quantum phase transitions has also many potential technological applications such as in memory storage devices or in processors for future quantum simulations. Technologies may be fabricated with materials showing interesting behaviour of charge, spin and current at cryogenic temperatures.

In this report, a Graphic User Interface will be presented which helps to visualize quantum critical scaling. Then, the GUI will be used to determine the critical scaling of the magnetization and thermal expansion of $Cu(C_4H_4N_2)(NO_3)_2$ as already done in [1]. A critical scaling of the susceptibility of $LiErF_4$ is found. Finally, there seems to be a critical scaling for the dielectric constant of $Ba_2CoGe_2O_7$ but stronger evidence is needed to confirm this hypothesis.

1 Theoretical background

1.1 Classical second order phase transitions

For many liquids, as for example water, the vapour pressure curve does not extend infinitely, it reaches some point called critical point. This critical point is characterized by a critical density, temperature and pressure. At that point, vapor and water do not coexist anymore. At that point the liquid state changes continuously to the vapor state. Apart from being the end of line in the PT phase diagram this point has some interesting properties. Close to that point, a small change in pressure makes the density vary a lot. Mathematically, this means that $\left(\frac{1}{\rho}\frac{\partial\rho}{\partial P}\right)_T$ which is called the compressibility K_T , is infinite at that point. Moreover, the difference between the liquid and gas densities ρ_l and ρ_g vanishes at the critical point. Finally, if the critical point is approached, the spatial correlations of the density difference fluctuations become non-zero at very large distances compared to the characteristic scale of the system (lattice parameter for example).

Such an example of phase transition is formally called a continuous phase transition. In such a transition, a thermodynamic potential has a second order derivative which is either continuous or infinite at the critical point. Another feature of a continuous phase transition is the order parameter, it is non-zero in the ordered phase and zero above the critical point. Finally the typical length scale ξ of the spatial correlations diverges as $\xi \propto |t|^{-\nu}$. t is the reduced temperature $\left(\frac{T-T_c}{T_c}\right)$ and ν is the correlation length critical exponent (see below in table 1).

There are also analogous long-range correlations in time close to the critical point. The fluctuations typically decay like $\tau_c \propto \xi^z \propto |t|^{-\nu z}$ where τ_c is the correlation (or equilibrium time). And what is quite important is that close to the critical point, there is no other characteristic timescale or length scale than τ_c and ξ respectively.

What is remarkable about the theory of phase transitions is that is applicable not only to the

Quantity	Exponent	Definition	Conditions
Specific heat C	α	$C \propto, \lvert t\rvert^{-\alpha}$	$t \longrightarrow 0$ and $B = 0$
Order parameter m	β	$m \propto (-t)^{\beta}$	$t \longrightarrow 0$ from below and $B = 0$
Susceptibility χ	γ	$\chi \propto \lvert t\rvert^{-\gamma}$	$t \longrightarrow 0$ and $B = 0$
Critical isotherm	δ	$B \propto \lvert m\rvert^{\delta}$ sign m	$B \longrightarrow 0$ and $t = 0$
Correlation length ξ	ν	$\xi \propto \lvert t\rvert^{-\nu}$	$t \longrightarrow 0$ and $B = 0$
Correlation function G	η	$G(r) \propto \lvert r\rvert^{-d+2-\eta}$	$t = 0$ and $B = 0$
Correlation time τ_c	z	$\tau_c \propto \xi^{z}$	$t \longrightarrow 0$ and $B = 0$

Table 1: Classical critical exponents for magnets [2]

fluid phase diagram of a given substance. The theory can also be applied for example in the case of the lattice gas-model of a ferromagnetic system. At some critical temperature T_c, this system changes from the ferromagnetic (ordered) to the paramagnetic phase (disordered). Quite clearly, temperature fluctuations which increase with bigger temperature are at the origin of this change of phase. In the fluid-magnetic analogy, the susceptibility $\chi_T = \left(\frac{\partial M}{\partial H}\right)_T$ takes the role of the compressibility. The order parameter is the magnetisation. The critical exponent of the correlation length ν equals approximately $\frac{1}{3}$ for the magnetic and the fluid systems.
In the study of magnets, there are actually plenty of critical exponents. See table 1.

As mentioned above, continuous phase transitions show a universal behavior on a variety of physical systems. What is also extraordinary is that the university classes of critical exponents depend only on the symmetries of the order parameter and on the space dimensionality of the system. The microscopic details of the Hamiltonian get unimportant close to the critical point. Thus, the critical exponents can be studied by exploring a simpler model Hamiltonian which belongs to the same university class.

1.2 Quantum phase transitions

So far, the discussion was about classical phase transitions, which means that the qualitative change in the system properties was driven only by thermal fluctuations. At $T = 0$ however, the thermal fluctuations are zero and another set of phase transitions can nevertheless occur. These phase transitions happening at zero temperature are called *quantum phase transitions* and come from Heisenberg's uncertainty principle. The discussion will now mainly be focused on magnetic systems such as magnets.
What distinguishes a classical second order phase transition from the quantum one is that the latter has also a characteristic energy scale which vanishes as the critical point is approached. In other words, the energy gap between the ground state and the first excited state of the system vanishes like $\Delta \propto J|r - r_c|^{z\nu}$ where Δ is the energy spectrum gap and J is the energy scale of a characteristic microscopic coupling. r is the control parameter used to tune the system through

the quantum phase transition. It could be for example the pressure applied to the solid or the strength of an external field. It is assumed here that the system is at zero temperature. There is also an issue about finiteness or infiniteness and nonanalyticity at $r = r_c$ of the system which will not be treated here.
The main point of interest is actually not the critical point where the quantum phase transition is happening but rather, the region above in the phase diagram, where there is an interplay of quantum and thermal fluctuations at finite temperature. One of the reasons for this is that it is extremely difficult or almost impossible to reach an absolute zero temperature experimentally. To go into further details, two different cases of phase diagrams showing a quantum critical region will be considered separately. In the first case, no long range order can exist at finite temperatures. In that case, there is no phase transition at finite temperature. However, there are so-called crossovers, which delimit the region of thermal fluctuations (left), the quantum critical region (middle) and quantum fluctuations (right). See figure 1. On top, the QCR stops when $k_B T$ reaches the typical exchange energy. On the right and on the left, the region is delimited by the condition $k_B T \propto |r - r_c|^{\nu z}$.

In the second case, long-range order can actually exist at finite temperature (see figure 1).

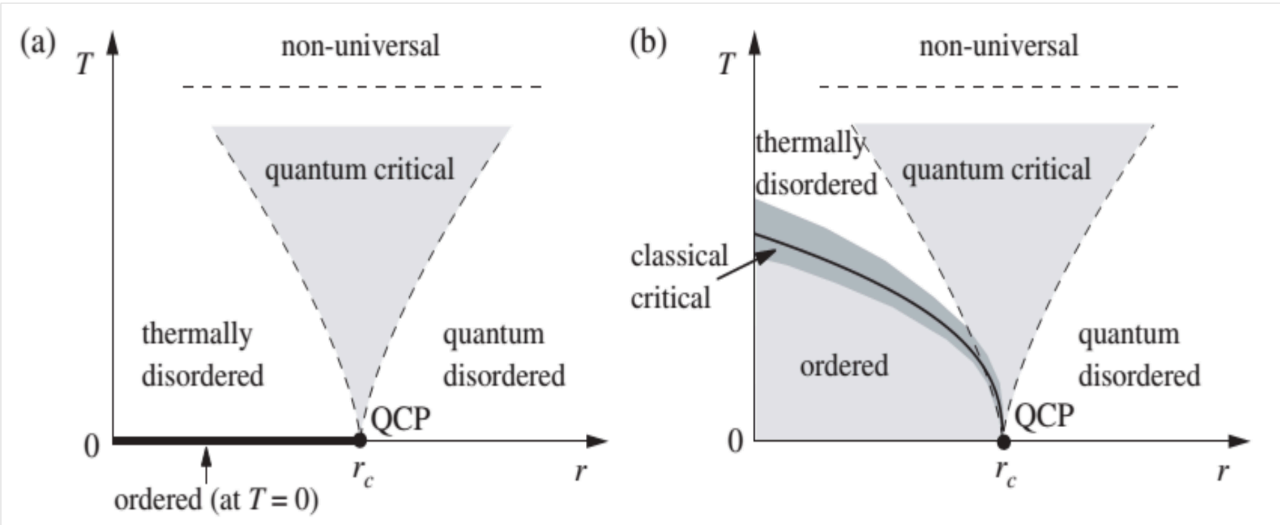


Figure 1: Schematic phase diagrams close to a quantum critical point [2].

What was said for the first case concerning the cutoff and the right and left limits remains true except that now, there is an ordered phase at finite temperature and a phase transition at finite temperature. Because of this, on the right of the phase transition, there will be a small region (which gets smaller with decreasing temperature) where the behaviour will be entirely classical instead of being quantum critical. To determine the limits of the quantum critical region more precisely, different energy scales have to be defined. The thermal energy is defined by $k_B T$. The typical energy of long-distance order parameter fluctuations is $\hbar\omega_c$. When the thermal energy is bigger than the energy of the long-range order parameter fluctuations, the system is driven by classical fluctuations and the considered point is not in the QCR. Knowing that, $\tau_c^{-1} \propto \hbar\omega_c \propto |t|^{\nu z}$ the behaviour is classical close to this phase transition. Indeed, t will be small

Quantity	Exponent	Definition	Conditions
Correlation length ξ	ν	$\xi \propto \lvert h - h_c \rvert^{-\nu}$	$h \longrightarrow h_c$ and $h_L = 0$
Order parameter m_x	β	$m_x \propto (h_c - h)^{\beta}$	$h \longrightarrow h_c$ from below and $h_L = 0$
Specific heat C	α	$C \propto \lvert h - h_c \rvert^{-\alpha}$	$h \longrightarrow h_c$ and $h_L = 0$
Susceptibility χ	γ	$C \propto, \lvert h - h_c \rvert^{-\gamma}$	$h \longrightarrow h_c$ and $h_L = 0$
Critical isotherm	δ	$h_L \propto \lvert m_x \rvert^{\delta}$ sign m_x	$h_L \longrightarrow 0$ and $h = h_c$
Correlation function G	η	$G(r) \propto \lvert r \rvert^{-d+2-\eta}$	$h = h_c$ and $h_L = 0$
Correlation time ξ_τ	z	$\xi_\tau \propto \xi^z$	$h \longrightarrow h_c$ and $h_L = 0$

Table 2: Quantum critical exponents for magnets [6]. h_L is the longitudinal applied magnetic field.

and so will be $\hbar\omega_c$. Therefore $|t|^{\nu z} < k_B T_c$ and thermal fluctuations will govern the system.
In the quantum critical region and close to the quantum critical point, the physics are dominated by thermal excitations of the quantum critical ground state and the behaviour is universal. Table 2 shows some quantum critical exponents associated with a quantum critical point in magnetic systems.
A concrete example of a second order quantum phase transition is now provided.

Consider the transverse Ising model on a hypercubic lattice of for example $LiHoF_4$. The Hamiltonian is given by :

$$H = -\sum_{<ij>} J_{ij}\sigma_i^x \sigma_j^x - h\sum_i \sigma_i^z \tag{1}$$

where σ_i^a, $a = x, y, z$ denote the Pauli spin matrices. h is here analogous to r which has been defined previously. When the first term of this Hamiltonian dominates over the second one, it means that the system is determined by magnetic dipolar interactions which cause all the spins to align in the same direction. The system is in the ferromagnetic state. As h increases however, the second term gains some importance and some spins will be flipped because of the interaction with the applied transverse magnetic field. Indeed, if the Pauli matrix σ_i^x acts on the eigenstate of σ_i^z with eigenvalue $+1$, the spin is flipped down. Therefore, because σ_i^x does not commute with σ_i^z, there will be a Heisenberg uncertainty relation and quantum fluctuations will appear. The system will thus change from the ferromagnetic state to the quantum paramagnet state.

2 Results

2.1 Scaling analysis for magnetization and thermal expansion of $Cu(C_4H_4N_2)(NO_3)_2$

First, the magnetization data of $Cu(C_4H_4N_2)(NO_3)_2$ taken from [1] was analysed. On figure 2, $(M_s - M)/H$ as a function of tempertature is depicted, as already done in [1]. On figure 3, with the help of the GUI, a collapse of all the data sets for the various applied magnetic fields was found for the scaling functions $y = (M_s - M)/T^\beta$ and $x = g\mu_B(H_s - H)/k_BT$. When $H < H_s$, only data belonging $T > T^*$ where T^* is defined such that $k_BT^* = 0.76328 g\mu_B(H_s - H)$ was selected for the scaling. H_s is the saturation magnetic field. $g = 2.265$. The values of H_s and β which minimize the χ^2 are $H_s = 14.01$ and $\beta = 0.47$. λ was set equal to 1. The χ^2 was obtained by computing for every scaling the best third order polynomial and then summing up all the squares of the difference of the measured data and the third order polynomial normalized by the number of points. Please see code for more details. One could also choose to compute χ^2 with respect to the theoretical function, but since most of time time the theoretical function is not known, it makes more sense to compare the data to a polynomial. The known theoretical function in this case is

$$M_s - M = g\mu_B \left(\frac{2k_BT}{J}\right)^\beta \mathcal{M}(\mu/k_BT) \tag{2}$$

and

$$\mathcal{M} = \frac{1}{\pi}\int_0^\infty \frac{1}{e^{x^2-\mu/k_BT}+1}dx \tag{3}$$

$\mu = g\mu_B(H_s - H)$.

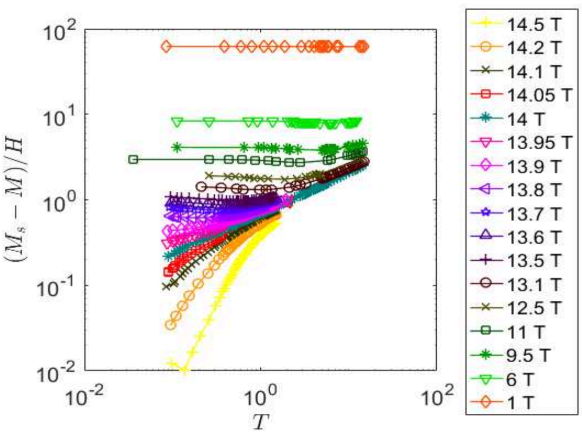


Figure 2: Raw data of $Cu(C_4H_4N_2)(NO_3)_2$

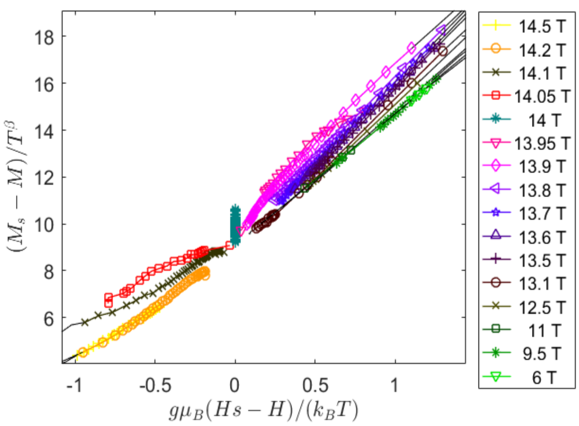


Figure 3: Quantum critical scaling of the magnetization

After that, the thermal expansion data of $Cu(C_4H_4N_2)(NO_3)_2$ taken from [1] was analysed. Figure 4 shows the right scaling with scaling functions $y = \alpha/T^\beta$ and $x = g\mu_B(H_s - H)/k_BT$. Taking the same boundaries as for the magnetization data, the optimal scaling was obtained for

$\lambda = 0.97$, $\beta = -0.506$ and $H_s = 13.88$. The χ^2 was again computed with respect to the best third order polynomial.

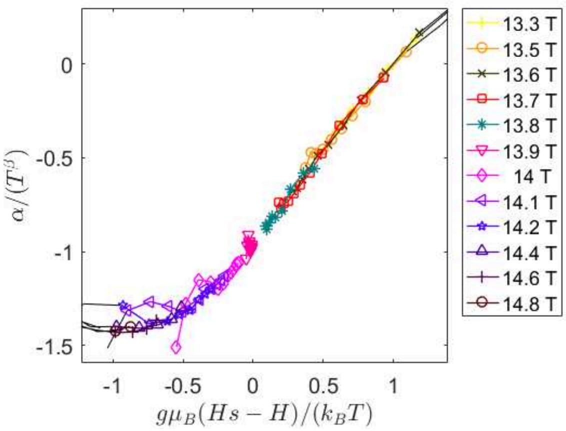


Figure 4: Quantum critical scaling of the thermal expansion

The aim of these two scaling analysis that were already done in [1] was really just to show that the GUI works properly and gives the same results as the mentioned paper.

2.2 Scaling analysis for susceptibility of $LiErF_4$

Then, the GUI was used to determine the scaling of the susceptibility data of $LiErF_4$ [5, 8]. The susceptibility data as a function of temperature and applied magnetic field is depicted in figure 5.
Various scaling functions shown in table 3 were tried and scaling was found for $y = (\chi - \chi_0)/T^{\beta}$ and $x = g\mu_B(H_s - H)/k_BT$ as shown in figure 6.

	x-axis	y-axis
1	$g\mu_B(H_s - H)/(k_BT)$	χ/T^{β}
2	$g\mu_B(H_s - H)/(k_BT)$	$H\chi/T^{\beta}$
3	$g\mu_B(H_s - H)/(k_BT)$	$(\chi - \chi_0)/T^{\beta}$
4	$g\mu_B(H_s - H)/(k_BT)$	$H(\chi - \chi_0)/T^{\beta}$

Table 3: Trial scaling functions for $LiErF_4$

Only data belonging to $T > T_p + 0.2T_p$ and such that $T > 2K$ was selected. T_p corresponds to the temperature value of each curve for which there is a peak in the susceptibility. This was a first guess obtained by considering the curves on figure 5, but nevertheless leads to scaling. The values of χ_0 and β which minimize the χ^2 are $\chi_0 = 0.08$ and $\beta = -0.398$. The following parameters were set : $\lambda = 1$ and $H_s = 0.37$.

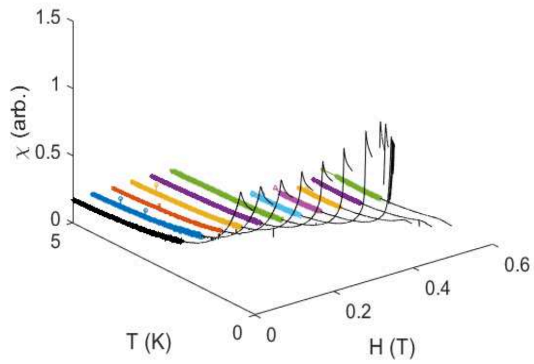


Figure 5: Raw data of $LiErF_4$. The data with non-black color is in the QCR and will be considered for the scaling.

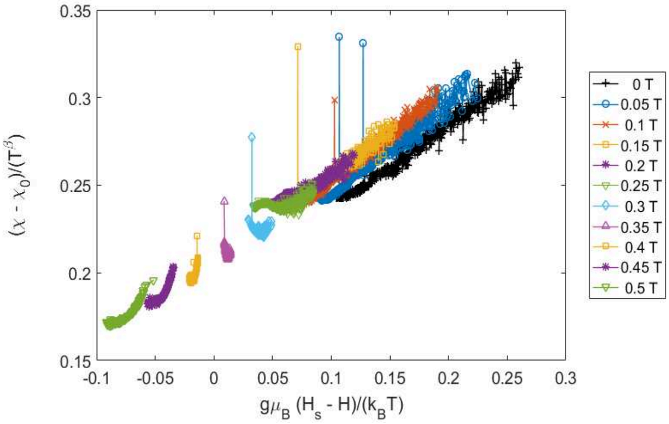


Figure 6: Quantum critical scaling of $LiErF_4$

2.3 Scaling analysis for the dielectric constant of $Ba_2CoGe_2O_7$

The GUI was finally used for the scaling of the dielectric constant of $Ba_2CoGe_2O_7$ [7]. In figure 7, the dielectric constant as a function of temperature and the applied magnetic field is plotted. Various scaling functions were tried as shown in table 4. For the scaling function $y = (\epsilon - \epsilon_0)/T^\beta$ and $x = g\mu_B(H_s - H)/k_BT$, it seems that there is an overlapping of the curves with $H = 43$, 44, 45 T as shown in figure 8. The parameter values are : $\beta = -0.5$ and $\epsilon_0 = 9.3$. The following parameters were set : $\lambda = 1$ and $H_s = 37.1$. Since there is not an overlapping for all the curves, the QCR should be defined more precisely to see if one can get a better overlapping. Then one could try to fit it to a third order polynomial and determine the optimal parameters.

	x-axis	y-axis
1	$g\mu_B(H_s - H)/(k_BT)$	ϵ/T^β
2	$g\mu_B(H_s - H)/(k_BT)$	$H\epsilon/T^\beta$
3	$g\mu_B(H_s - H)/(k_BT)$	$(\epsilon - \epsilon_0)/T^\beta$
4	$g\mu_B(H_s - H)/(k_BT)$	$H(\epsilon - \epsilon_0)/T^\beta$

Table 4: Trial scaling functions for $Ba_2CoGe_2O_7$

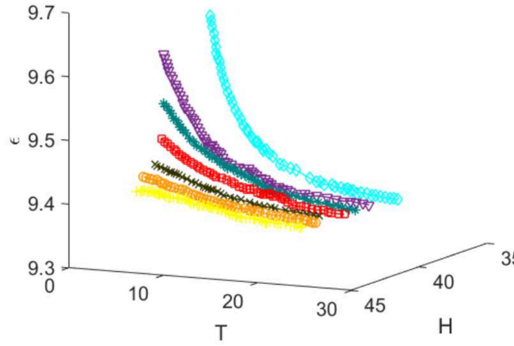


Figure 7: Raw data of $Ba_2CoGe_2O_7$.

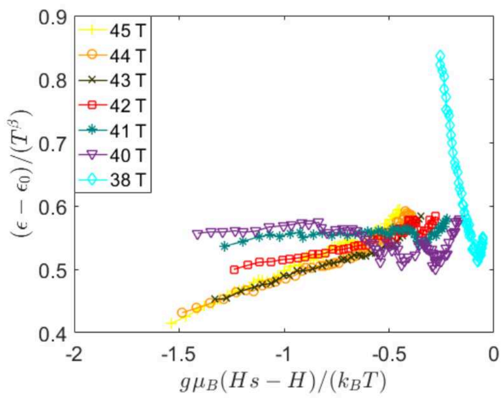


Figure 8: Quantum critical scaling of $Ba_2CoGe_2O_7$.

3 Code description

In figures 9, 10 and 11 there is an explanation for the code used and how the different functions are related to each other. For more details, the reader should read through the code given below.

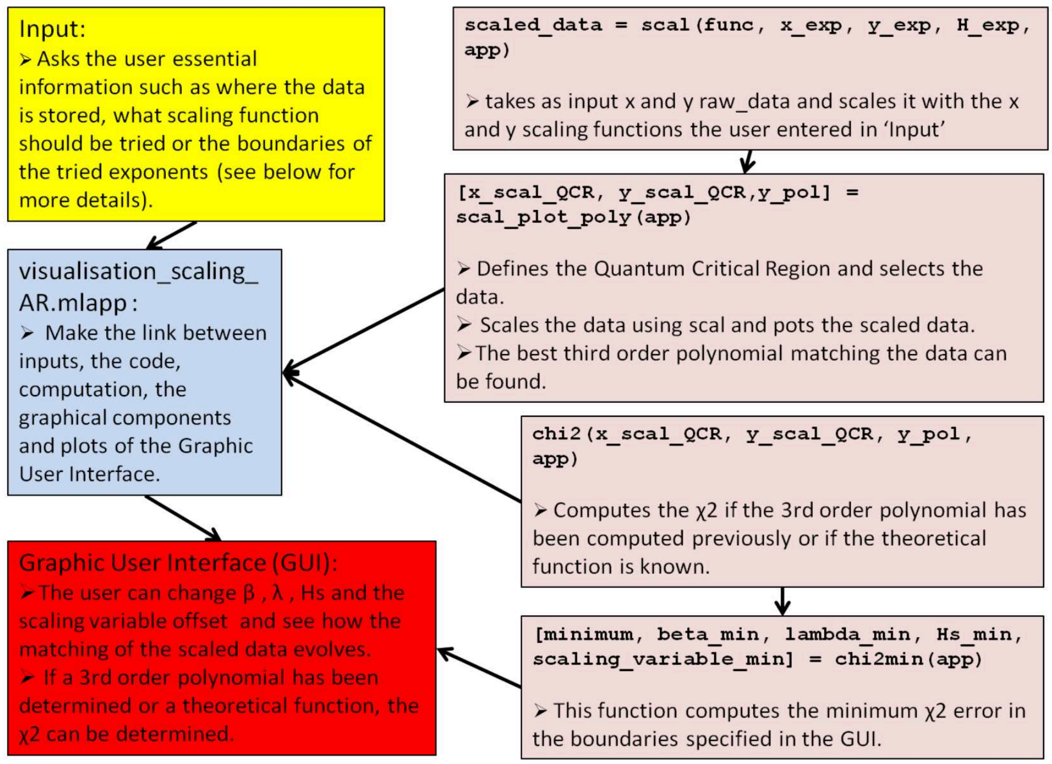


Figure 9: Links between the MATLAB files used to create the Graphic User Interface

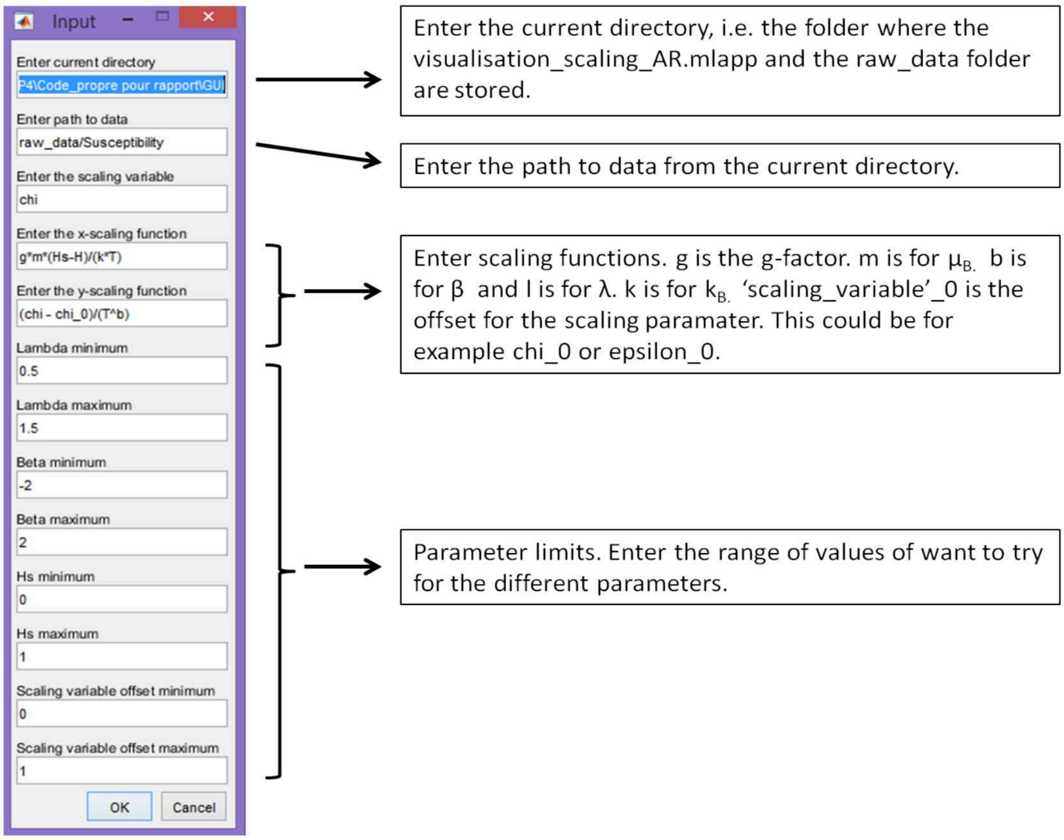
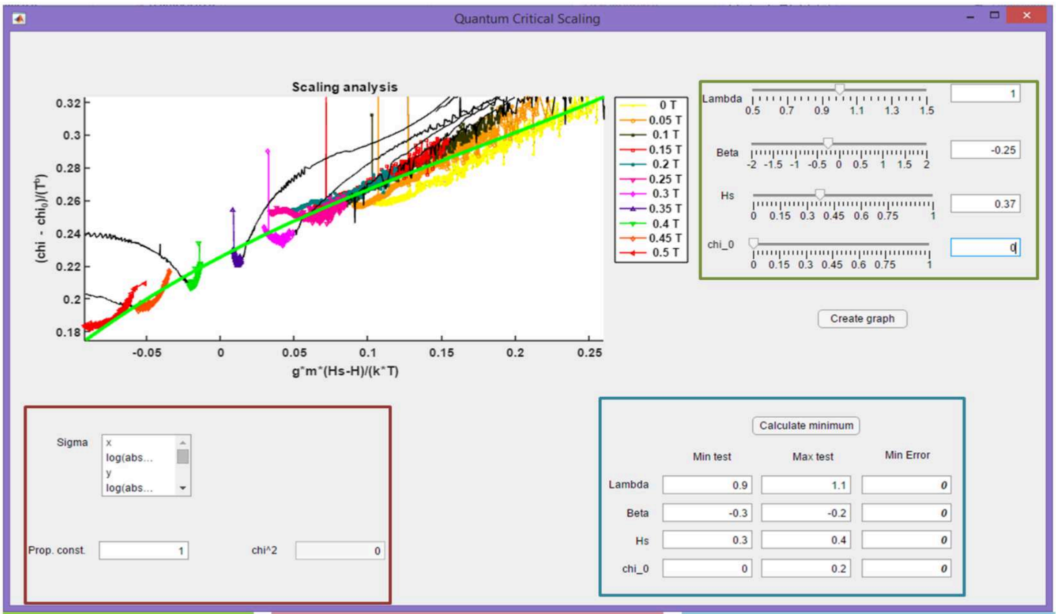


Figure 10: Description of the 'Input' window



First the user can change the 4 parameters and see for which values there seems to be a scaling.

Then the sigma and the proportionality constant can be defined. The exact meaning of the sigma can be found in the code comments of chi2.m. The χ2 error is then computed.

Finally, the user can determine the λ, β, Hs and scaling variable, such that the χ2 error is minimal after having entered the interval of parameters which he wants to test.

Figure 11: Description of how to use the GUI

4 Useful information about the GUI

1. To run the GUI, the user must double-click the *visualization scaling AR* (*.mlapp*) and then fill in the 'Input' window which pops up. After clicking 'ok' the GUI will appear. The *.mlapp* file can only be read by MATLAB 2016 and later versions.

2. Make sure that the parameter values entered are in the boundaries specified in the 'Input'.

3. Once you click on the 'Create graph' button, you cannot use the GUI anymore, you have to restart it.

4. The calculation of the minimum error is really long because each time there is a change in parameter, the best 3rd order polynomial has to be computed. To avoid too long computation time, set constant at least one of the four parameters. If the maximum value is not equal to the minimum value for a given parameter, the increment at each iteration will be one tenth of the difference between the minimum and the maximum value. This amounts to 1000 iterations if one of the parameters is set constant.

5. If you want to change the boundaries of the considered quantum critical region, you can do so at line 91 in the *scal plot poly* function.

5 Code

5.1 The scal plot poly function

```
1  function [x_scal_QCR,y_scal_QCR, y_pol] = scal_plot_poly(app)
2
3          % File to define colors and markers for plots
4          load ColorMarker.mat;
5
6          % Clear the axes on the app
7          cla(app.UIAxes)
8
9          % Rename the app components
10         beta = app.BetaSlider.Value;
11         lambda = app.LambdaSlider.Value;
12         Hs = app.HsSlider.Value;
13         epsilon_0 = app.ScVarSlider.Value;
14
15         % Initialize the H vector which will be used for legends in
16         % graphs.
17         H = 0;
18
19         %===================================================
20         % Load and order data %
21         % The input data should be a .dat file with first column
22         % temperature and second column the parameter that one wants to
```

```
        % scale.
        % The applied magnetic field is in the first line second column
        % The data starts at the third line.
        % DATA is a struct with initial field names, folder, date, bytes,
            isdir, datenum.
        % In the following lines x_data, y_data, H will be added to the
            struct DATA where x_data, y_data and H come from
        % the values in the different data sets in the app.currentpathtodata.
        DATA = dir(app.currentpathtodata);

        % Initialization
        [DATA(:).x_data] = deal(randn(5,1));
        [DATA(:).y_data] = deal(randn(6,1));
        [DATA(:).H] = deal(randn(7,1));

        % i starts at 3 because the two first datas are irrelevant.
        for i = 3:length(DATA)
                filename = strcat(app.currentpathtodata,'/',DATA(i).name);
                T = readtable(filename,'HeaderLines',2);
                DATA(i).x_data = T(:,1);
                DATA(i).y_data = T(:,2) ;
                DATA(i).H = dlmread(filename,'','B1..B1');
                % H is also defined as a vector. It will be used in the
                % legends for graphs.
                H(i-2) = dlmread(filename,'','B1..B1');
        end



        %=================================================================%
        % Quantum Critical Scaling analysis

        % 1) Select data in quantum critical region(QCR)

        % Initialize the indices which corresponding data is in the QCR
        indices = cell(length(DATA)-2,1);

            % a) Define the boundaries of the QCR

            % for the magnetization and thermal
            % expansion data of the paper " Quantum critical scaling for a
            % Heisenberg spin-(1/2) chain around saturation"
            % These boundaries might change for other magnets
            T_star = 0.76328*app.g*app.muB/app.kB.*(Hs - [DATA(3:length(DATA)
                ).H]);
            Delta = app.g*app.muB/app.kB.*([DATA(3:length(DATA)).H] - Hs);
            upper_bound = 10.3;

            % Sometimes it is useful to know which data corresponds to
            % a peak (phase change or crossover), in order to select the
```

```
        points just above these
        % peaks in the phase diagram for example.
        M = 0;
        I = 0;
        maxim = 0;
        for i = 3:length(DATA)
            [M(i) I(i)]= max([DATA(i).y_data{:,1}]);
            maxim(i-2) = DATA(i).x_data{I(i),1};
        end



        % b) Find the data in the QCR



        H_bool = 0; % This vector contains 1 if for  a given data set there is at least one point in the QCR.
                    % Otherwise is it 0 (see below line 106).

        % Find the indices for which the corresponding data is in the QCR.
        for i = 3:length(DATA)

                % For the magnetization and thermal expansion data
                if DATA(i).H < Hs

                indices{i-2,1} = find( [DATA(i).x_data{:,1}] > T_star(i-2)   &   [DATA(i).x_data{:,1}] < 10.3);

                else

                indices{i-2,1} = find( [DATA(i).x_data{:,1}] > Delta(i-2)   &   [DATA(i).x_data{:,1}] < 10.3);
                end

                % For the LiErF4 magnet
                 %indices{i-2,1} = find( [DATA(i).x_data{:,1}] > (0.2*maxim(i-2) + maxim(i-2)) & [DATA(i).x_data{:,1}] > 2.0);



                [rr ff] = size(indices{i-2,1});
                if rr > 0
                    H_bool(i-2) = 1;
                else
                    H_bool(i-2) = 0;
                end
        end
```

```
            % indices_H_red gives the indices of H for which at least one point of the
            % corresponding x, y_data is in the QCR
            indices_H_red = find(H_bool == 1);




            % Select the data in the QCR
            x_QCR = cell(length(DATA)-2,1);
            y_QCR = cell(length(DATA)-2,1);
            for i = 3:length(DATA)
                x_QCR{i-2} = [DATA(i).x_data{indices{i-2,1},1}];
                y_QCR{i-2} = [DATA(i).y_data{indices{i-2,1},1}];
            end
            % All the data (will be useful for the graphs)
            for i = 3:length(DATA)
                H_exp{i-2,1}= DATA(i).H;
                x_all{i-2,1}= [DATA(i).x_data{:,1}];
                y_all{i-2,1}= [DATA(i).y_data{:,1}];
            end

        % 2) Scale the variables
        % scal() scales the variables according to the function that
        % the user enters.
        x_scal_QCR = scal(app.xfunction, x_QCR, y_QCR, H_exp, app);
        y_scal_QCR = scal(app.yfunction, x_QCR, y_QCR, H_exp, app);

        %  Scale all the variables.
        x_scal_all = scal(app.xfunction, x_all, y_all, H_exp, app);
        y_scal_all = scal(app.yfunction, x_all, y_all, H_exp, app);


        % 3) Plot the Scaled variables in the QCR and outside the QCR for comparison.
        % H_red and hh will be used in the legends of the plots.
        H_red = num2str(transpose(H(indices_H_red)));
        hh = gobjects(length(H_red)-2,1);
        bbb = 0;
        % Plot scaled variables
        if app.c ~= 1 % to check that the program is not computing the minimum chi2. Otherwise the program would make appear a graph at which iteration which would be a loss of time.
            for i = 1:length(DATA)-2
                plot(app.UIAxes, x_scal_all{i}, y_scal_all{i},'-k','DisplayName', 'off')
                if H_bool(i) == 1
                    bbb = bbb + 1;
                    hold(app.UIAxes,'on')
                    hh(bbb) = plot(app.UIAxes,x_scal_QCR{i}, y_scal_QCR{i},'
```

```
                    Marker', mkr{i}, 'Color', clr(i,:), 'MarkerSize', 3, 'LineWidth', 1);
                end
            end
            legend(app.UIAxes,hh,strcat(H_red,' T'),'Location','northeastoutside')
        end






        % 'Create graph button'
        % if app.a = 1, it means that the user has pressed the 'create
        % graph' button. The graph that is seen on the graphic user
        % interface will pop up in figure 1 as shown here.

        if app.a == 1
            figure(1)
                gg = gobjects(length(H_red)-2,1);
                bbbb = 0;
                for i = 1:length(DATA)-2

                    plot(x_scal_all{i}, y_scal_all{i},'-k','DisplayName', 'off');
                    if H_bool(i) == 1
                        bbbb = bbbb + 1;
                        hold on
                        gg(bbbb) = plot(x_scal_QCR{i}, y_scal_QCR{i},'Marker', mkr{i}, 'Color', clr(i,:), 'MarkerSize', 7, 'LineWidth', 1);
                    end
                end

        legend(gg,strcat(H_red,' T'),'Location','northeastoutside')

        % The following lines change the writing of the labels to the
        % Latex Interpreter language
        app.xfunction = strrep(app.xfunction,'k','k_B');
        app.xfunction = strrep(app.xfunction,'m','\mu_B');
        app.xfunction = strrep(app.xfunction,'l','\lambda');
        app.xfunction = strrep(app.xfunction,'*','');
        app.yfunction = strrep(app.yfunction,'b','{\beta}');
        app.yfunction = strrep(app.yfunction,'*','');
        app.yfunction = strrep(app.yfunction,'alpha',' \alpha ');
        app.yfunction = strrep(app.yfunction,'epsilon',' \epsilon ');
        app.yfunction = strrep(app.yfunction,'chi',' \chi ');
        app.yfunction = strrep(app.yfunction,'xi',' \xi ');
        xlabel(strcat('$',app.xfunction,'$'),'Interpreter','latex')
```

```
            ylabel(strcat('$',app.yfunction,'$'),'Interpreter','latex')
            set(gca,'fontsize',15)

            end

            % Collapse the scaled data in the QCR into one single big
            % vector, needed for the calculation of the best third order
            % polynomial and the chi2
             x_scal_QCR_all = x_scal_QCR{1,:};
             y_scal_QCR_all = y_scal_QCR{1,:};
             for i = 2:length(DATA) - 2
                 x_scal_QCR_all = vertcat(x_scal_QCR_all,x_scal_QCR{i,:});
                 y_scal_QCR_all = vertcat(y_scal_QCR_all,y_scal_QCR{i,:});
             end
             x_scal_QCR_all_sort = sort(x_scal_QCR_all);


            % Best third order polynomial

            p = polyfit(x_scal_QCR_all,y_scal_QCR_all,3);
            y_pol_sort = polyval(p,x_scal_QCR_all_sort);
            y_pol = polyval(p,x_scal_QCR_all);
            if app.c ~= 1
                h = plot(app.UIAxes, x_scal_QCR_all_sort, y_pol_sort,'-g', '
                    LineWidth', 3, 'DisplayName', '3rd order poly.');
            end
            % legend(app.UIAxes, [h],{'Best 3rd order poly.'},'Location','
                Northwest');
            hold(app.UIAxes, 'off');
            % If needed : you can set the xlimits and ylimits manually here
            app.UIAxes.XLim = [min(x_scal_QCR_all_sort)    max(x_scal_QCR_all_sort
                )];
            app.UIAxes.YLim = [min(y_pol_sort)    max(y_pol_sort)];


end
```

5.2 The scal function

```
% This function scales the data
function scaled_data = scal(func, x_exp, y_exp, H_exp, app)
    % To ensure vector element mulitiplication
    func = insertBefore(func,'*','.');
    func = insertBefore(func,'/','.');
    func = insertBefore(func,'^','.');
    % Substitute the scaling variable by y and the scaling offset by
    % epsilon_0
    func = strrep(func,strcat(app.scaling_variable,'_0'),'epsilon_0');
    func = strrep(func,app.scaling_variable,'y');

    func_str = strcat('@(T,y,b,l,Hs,H,g,m,k,epsilon_0)', func);
    f = str2func(func_str);
    b = app.BetaSlider.Value;
    l = app.LambdaSlider.Value;
    Hs = app.HsSlider.Value;
    epsilon_0 = app.ScVarSlider.Value;
    g = app.g;
    m = app.muB;
    k = app.kB;
    scaled_data = cellfun(@(T,y,H) f(T, y, b,l,Hs,H,g,m,k,epsilon_0), x_exp,
        y_exp, H_exp,'UniformOutput', false);

end
```

5.3 The chi2 function

```
% This chi2 function computes the chi2 error taking into account the error
% between the best third order polynomial and the scaled data.
function chi2(x_scal_QCR, y_scal_QCR, y_pol, app)

            % Determination of sigma. Sigma can be chosen in the GUI to be
            % proportional to x, log(x), y, log(y), etc. Again x
            % correponds to temperature and y is the scaled data.
            % Therefore with sigma it will be possible to weight the
            % error. If for example the user wants to weight a lot values
            % which have small x, he can set the sigma to 'x'. If the
            % 'None' option is ticked, there is no weighting and simply all
            % the errors squared are added
            % The proportionality factor can also be chosen in the GUI.
             if strcmp(app.SigmaListBox.Value,'None') ~= 1
                func_str = strcat('@(x,y)', app.SigmaListBox.Value);
                f = str2func(func_str);
                sigma = cellfun(f, x_scal_QCR, y_scal_QCR, 'UniformOutput',
                    false);
            end

                   erreur_sd_3rdpoly = zeros(length(y_scal_QCR),1);

                   b = 0.0;
                   % This loop runs on all data sets.
                   for i = 1 : length(y_scal_QCR)
                       % This loop runs on all points in a given data set.
                       for j = 1 : length(x_scal_QCR{i,:})
                           a = b + j;
                           % This 'if' condition is here to avoid
                           % divisions by 0.
                           if strcmp(app.SigmaListBox.Value,'None') == 1
                               sigma{i,:}(j)= 1;
                           end
                           if sigma{i,:}(j)~= 0
                                 % Sum of the error scaled by sigma for a
                                 % given data set.
                                 erreur_sd_3rdpoly(i) = erreur_sd_3rdpoly(i) +
                                     ((y_scal_QCR{i,:}(j) - y_pol(a))^2)/(app.
                                     PropconstEditField.Value*sigma{i,:}(j))^2;
                           end
                       end
                       b = b + length(x_scal_QCR{i,:});

                   end
               if b ~= 0
                   app.chi2EditField.Value = sum(erreur_sd_3rdpoly)/b;
               else
                   app.chi2EditField.Value = 0;
```

```
                    end


end
```

5.4 The chi2min function

```
% This function computes the minimum chi2 error in the boundaries specified
% in the GUI.

function [minimum, beta_min, lambda_min, Hs_min, scaling_variable_min] = chi2min(
    app)
    lambda_int = (app.Lambda_M.Value - app.Lambda_m.Value)/10.0;
    beta_int = (app.Beta_M.Value - app.Beta_m.Value)/10.0;
    Hs_int = (app.Hs_M.Value - app.Hs_m.Value)/10.0;
    scaling_variable_int = (app.Scalingvariable_M.Value - app.Scalingvariable_m.
        Value)/10.0;
    minimum = 1000000000;
    beta_min = app.Beta_m.Value;
    lambda_min = app.Lambda_m.Value;
    Hs_min = app.Hs_m.Value;
    scaling_variable_min = app.Scalingvariable_m.Value;
     for i1 = app.Beta_m.Value : beta_int: app.Beta_m.Value
         app.BetaSlider.Value = i1;
        for i2 = app.Lambda_m.Value : lambda_int: app.Lambda_M.Value
            app.LambdaSlider.Value = i2;
            for i3 = app.Hs_m.Value : Hs_int: app.Hs_M.Value
                app.HsSlider.Value = i3;
                for i4 = app.Scalingvariable_m.Value:scaling_variable_int:app.
                    Scalingvariable_M.Value
                    app.ScalingvariableoffsetSlider.Value = i4;
                    [x_scal_QCR, y_scal_QCR, y_pol] = scal_plot_poly(app);
                    chi2(x_scal_QCR, y_scal_QCR, y_pol, app);
                    if app.chi2EditField.Value <= minimum
                        minimum = app.chi2EditField.Value;
                        beta_min = app.BetaSlider.Value;
                        lambda_min = app.LambdaSlider.Value;
                        Hs_min = app.HsSlider.Value;
                        scaling_variable_min = app.ScalingvariableoffsetSlider.
                            Value;
                    end
                end
            end
        end
    end

end
```

5.5 The visualisation scaling app (mlapp file)

```
classdef visualisation_scaling_AR < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                        matlab.ui.Figure
        UIAxes                          matlab.ui.control.UIAxes
        BetaSliderLabel                 matlab.ui.control.Label
        BetaSlider                      matlab.ui.control.Slider
        LambdaSliderLabel               matlab.ui.control.Label
        LambdaSlider                    matlab.ui.control.Slider
        chi2EditFieldLabel              matlab.ui.control.Label
        chi2EditField                   matlab.ui.control.NumericEditField
        HsSliderLabel                   matlab.ui.control.Label
        HsSlider                        matlab.ui.control.Slider
        SigmaListBoxLabel               matlab.ui.control.Label
        SigmaListBox                    matlab.ui.control.ListBox
        PropconstEditFieldLabel         matlab.ui.control.Label
        PropconstEditField              matlab.ui.control.NumericEditField
        ScVarSliderLabel                matlab.ui.control.Label
        ScVarSlider                     matlab.ui.control.Slider
        CreategraphButton               matlab.ui.control.Button
        LambdaEditField_2Label          matlab.ui.control.Label
        Lambda_m                        matlab.ui.control.NumericEditField
        Lambda_M                        matlab.ui.control.NumericEditField
        Beta_M                          matlab.ui.control.NumericEditField
        BetaEditField_2Label            matlab.ui.control.Label
        Beta_m                          matlab.ui.control.NumericEditField
        Hs_M                            matlab.ui.control.NumericEditField
        HsEditField_2Label              matlab.ui.control.Label
        Hs_m                            matlab.ui.control.NumericEditField
        Scalingvariable_M               matlab.ui.control.NumericEditField
        ScalingvariablemMLabel          matlab.ui.control.Label
        Scalingvariable_m               matlab.ui.control.NumericEditField
        MintestLabel                    matlab.ui.control.Label
        MaxtestLabel                    matlab.ui.control.Label
        lambda_min                      matlab.ui.control.NumericEditField
        beta_min                        matlab.ui.control.NumericEditField
        Hs_min                          matlab.ui.control.NumericEditField
        scaling_variable_min            matlab.ui.control.NumericEditField
        MinErrorLabel                   matlab.ui.control.Label
        CalculateminimumButton          matlab.ui.control.Button
        LambdaEditField                 matlab.ui.control.NumericEditField
        BetaEditField                   matlab.ui.control.NumericEditField
        HsEditField                     matlab.ui.control.NumericEditField
        ScalingvariableoffsetEditField  matlab.ui.control.NumericEditField
    end


```

```
    properties (Access = public)
            kB  = 1.3806485e-23;  % app.J/K
            muB = 9.2740099e-24;  % app.J/T
            g = 2.1;
            J = 10.3*1.3806485e-23;
            scaling_variable;
            xfunction;
            yfunction;
            currentpathtodata;
            minlambda;
            maxlambda;
            minbeta;
            maxbeta;
            minHs;
            maxHs;
            minoffset
            maxoffset
            a = 0; % if a = 1 it means that the user has pushed the 'Create graph' button.
            c = 0; % if 1 it means that the user wants to compute the minimum error and this c will ensure that in scal_plot_poly
              % no graph will appear at each iteration.
    end




    methods (Access = private)

        % Code that executes after component creation
        function startupFcn(app)
              % Input
              prompt = {'Enter current directory','Enter path to data ','Enter the scaling variable', 'Enter the x-scaling function', 'Enter the y-scaling function', 'Lambda minimum', 'Lambda maximum', 'Beta minimum', 'Beta maximum', 'Hs minimum', 'Hs maximum','Scaling variable offset minimum', 'Scaling variable offset maximum'};
              dlg_title = 'Input';
              num_lines = 1;
              defaultans = {'C:\Users\Annina Riedhauser\Documents\Master 1er Semestre\TP4\Code_propre pour rapport\GUI', 'raw_data/Susceptibility','chi','g*m*(Hs-H)/(k*T)','(chi - chi_0)/(T^b)', '0.5', '1.5', '-2', '2', '0', '1','0','1'};
              answer = inputdlg(prompt,dlg_title,num_lines,defaultans);

              cd(answer{1});
              app.currentpathtodata = answer{2};
              app.scaling_variable = answer{3};
```

```
            app.xfunction = answer{4};
            app.yfunction = answer{5};
            app.minlambda = answer{6};
            app.maxlambda = answer{7};
            app.minbeta = answer{8};
            app.maxbeta = answer{9};
            app.minHs = answer{10};
            app.maxHs = answer{11};
            app.minoffset = answer{12};
            app.maxoffset = answer{13};
            app.LambdaSlider.Limits = [str2double(app.minlambda) str2double(app.maxlambda)];
            app.BetaSlider.Limits = [str2double(app.minbeta) str2double(app.maxbeta)];
            app.HsSlider.Limits = [str2double(app.minHs) str2double(app.maxHs)];
            app.ScVarSlider.Limits = [str2double(app.minoffset) str2double(app.maxoffset)];
            app.ScVarSliderLabel.Text = strcat(app.scaling_variable, '_0');
            app.ScalingvariablemMLabel.Text = strcat(app.scaling_variable, '_0');
            xlabel(app.UIAxes, app.xfunction)
            ylabel(app.UIAxes, app.yfunction)
            title(app.UIAxes,'Scaling analysis')



            fig = app.UIFigure;
            name = fig.Name;
            fig.Name = 'Quantum Critical Scaling';

        end

        % Value changed function: BetaSlider
        function BetaSliderValueChanged(app, event)

            [x_scal_QCR, y_scal_QCR, y_pol] = scal_plot_poly(app);
           app.BetaEditField.Value = app.BetaSlider.Value;
        end

        % Value changed function: LambdaSlider
        function LambdaSliderValueChanged(app, event)

            [x_scal_QCR, y_scal_QCR, y_pol] = scal_plot_poly(app);
            app.LambdaEditField.Value = app.LambdaSlider.Value;
        end

        % Value changed function: HsSlider
        function HsSliderValueChanged(app, event)

```

```
            [x_scal_QCR, y_scal_QCR, y_pol] = scal_plot_poly( app);
            app.HsEditField.Value = app.HsSlider.Value;
        end

        % Value changed function: SigmaListBox
        function SigmaListBoxValueChanged(app, event)

            value = app.SigmaListBox.Value;
            [x_scal_QCR, y_scal_QCR, y_pol] = scal_plot_poly( app);
            chi2(x_scal_QCR, y_scal_QCR, y_pol, app);
        end

        % Value changed function: PropconstEditField
        function PropconstEditFieldValueChanged(app, event)

            value = app.PropconstEditField.Value;
            [x_scal_QCR, y_scal_QCR, y_pol] = scal_plot_poly( app);
            chi2(x_scal_QCR, y_scal_QCR, y_pol, app);
        end

        % Value changed function: LambdaEditField
        function LambdaEditFieldValueChanged(app, event)
            value = app.LambdaEditField.Value;
            app.LambdaSlider.Value = value
            [x_scal_QCR, y_scal_QCR, y_pol] = scal_plot_poly(app)

        end

        % Value changed function: BetaEditField
        function BetaEditFieldValueChanged(app, event)

            value = app.BetaEditField.Value;
            app.BetaSlider.Value = value
            [x_scal_QCR, y_scal_QCR, y_pol] = scal_plot_poly(app)

        end

        % Value changed function: HsEditField
        function HsEditFieldValueChanged(app, event)
            value = app.HsEditField.Value;
            app.HsSlider.Value = value;

            [x_scal_QCR, y_scal_QCR, y_pol] = scal_plot_poly(app);

        end

        % Value changed function: ScVarSlider
        function ScVarSliderValueChanged(app, event)

            [x_scal_QCR, y_scal_QCR, y_pol] = scal_plot_poly(app);
```

```matlab
            app.ScalingvariableoffsetEditField.Value = app.ScVarSlider.Value;


        end

        % Button pushed function: CreategraphButton
        function CreategraphButtonPushed(app, event)
             ff = figure(1)
             app.a = 1;
             [x_scal_QCR, y_scal_QCR, y_pol] = scal_plot_poly(app);

             app.a = 0 ;
        end

        % Callback function
        function ScalingvariableoffsetEditFieldValueChanged(app, event)
            value = app.ScalingvariableoffsetEditField.Value;
            app.ScVarSlider.Value = value
            [x_scal_QCR, y_scal_QCR, y_pol] = scal_plot_poly(app)
        end

        % Button pushed function: CalculateminimumButton
        function CalculateminimumButtonPushed(app, event)
            app.c = 1;
            [minimum, app.beta_min.Value, app.lambda_min.Value, app.Hs_min.Value,
                 app.scaling_variable_min.Value] = chi2min(app);
            app.c = 0;
        end
    end

    % App initialization and construction
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Create UIFigure
            app.UIFigure = uifigure;
            app.UIFigure.Position = [100 100 1154 681];
            app.UIFigure.Name = 'UI Figure';
            setAutoResize(app, app.UIFigure, true)

            % Create UIAxes
            app.UIAxes = uiaxes(app.UIFigure);
            title(app.UIAxes, 'Title');
            xlabel(app.UIAxes, 'X');
            ylabel(app.UIAxes, 'Y');
            app.UIAxes.PlotBoxAspectRatio = [1 0.5 0.5];
            app.UIAxes.PlotBoxAspectRatioMode = 'manual';
            app.UIAxes.FontSize = 14;
```

```
            app.UIAxes.FontWeight = 'bold';
            app.UIAxes.ColorOrder = [1 0 0;0 1 0;1 0 1;0 0 1;0.1 0.6 0.1;0.6 0.6
                0.6;0 0 0];
            app.UIAxes.LineStyleOrder = {'+-'; 'x-'; '*-'; 'd-'; 'o-'; 's-'; '-<'
                };
            app.UIAxes.Position = [16 203 746 514];

            % Create BetaSliderLabel
            app.BetaSliderLabel = uilabel(app.UIFigure);
            app.BetaSliderLabel.HorizontalAlignment = 'right';
            app.BetaSliderLabel.Position = [783 531 30 15];
            app.BetaSliderLabel.Text = 'Beta';

            % Create BetaSlider
            app.BetaSlider = uislider(app.UIFigure);
            app.BetaSlider.Limits = [0 1];
            app.BetaSlider.ValueChangedFcn = createCallbackFcn(app,
                @BetaSliderValueChanged, true);
            app.BetaSlider.Position = [827 547 195 3];

            % Create LambdaSliderLabel
            app.LambdaSliderLabel = uilabel(app.UIFigure);
            app.LambdaSliderLabel.HorizontalAlignment = 'right';
            app.LambdaSliderLabel.Position = [767 595 50 15];
            app.LambdaSliderLabel.Text = 'Lambda';

            % Create LambdaSlider
            app.LambdaSlider = uislider(app.UIFigure);
            app.LambdaSlider.Limits = [0 2];
            app.LambdaSlider.ValueChangedFcn = createCallbackFcn(app,
                @LambdaSliderValueChanged, true);
            app.LambdaSlider.Position = [828 611 194 3];
            app.LambdaSlider.Value = 1;

            % Create chi2EditFieldLabel
            app.chi2EditFieldLabel = uilabel(app.UIFigure);
            app.chi2EditFieldLabel.HorizontalAlignment = 'right';
            app.chi2EditFieldLabel.Position = [265 59 34 15];
            app.chi2EditFieldLabel.Text = 'chi^2';

            % Create chi2EditField
            app.chi2EditField = uieditfield(app.UIFigure, 'numeric');
            app.chi2EditField.Editable = 'off';
            app.chi2EditField.Position = [319 55 100 22];

            % Create HsSliderLabel
            app.HsSliderLabel = uilabel(app.UIFigure);
            app.HsSliderLabel.HorizontalAlignment = 'right';
            app.HsSliderLabel.Position = [783 480 25 15];
            app.HsSliderLabel.Text = 'Hs';
```

```
            % Create HsSlider
            app.HsSlider = uislider(app.UIFigure);
            app.HsSlider.Limits = [0.2 0.6];
            app.HsSlider.ValueChangedFcn = createCallbackFcn(app,
                @HsSliderValueChanged, true);
            app.HsSlider.Position = [829 486 200 3];
            app.HsSlider.Value = 0.4;

            % Create SigmaListBoxLabel
            app.SigmaListBoxLabel = uilabel(app.UIFigure);
            app.SigmaListBoxLabel.HorizontalAlignment = 'right';
            app.SigmaListBoxLabel.Position = [48 187 40 15];
            app.SigmaListBoxLabel.Text = 'Sigma';

            % Create SigmaListBox
            app.SigmaListBox = uilistbox(app.UIFigure);
            app.SigmaListBox.Items = {'x', 'log(abs(x))', 'y', 'log(abs(y))', '1/
                x', '1/log(abs(x))', '1/y', '1/log(abs(y))', 'None'};
            app.SigmaListBox.ValueChangedFcn = createCallbackFcn(app,
                @SigmaListBoxValueChanged, true);
            app.SigmaListBox.Position = [103 130 100 74];
            app.SigmaListBox.Value = 'None';

            % Create PropconstEditFieldLabel
            app.PropconstEditFieldLabel = uilabel(app.UIFigure);
            app.PropconstEditFieldLabel.HorizontalAlignment = 'right';
            app.PropconstEditFieldLabel.Position = [16 59 69 15];
            app.PropconstEditFieldLabel.Text = 'Prop. const.';

            % Create PropconstEditField
            app.PropconstEditField = uieditfield(app.UIFigure, 'numeric');
            app.PropconstEditField.ValueChangedFcn = createCallbackFcn(app,
                @PropconstEditFieldValueChanged, true);
            app.PropconstEditField.Position = [100 55 100 22];
            app.PropconstEditField.Value = 1;

            % Create ScVarSliderLabel
            app.ScVarSliderLabel = uilabel(app.UIFigure);
            app.ScVarSliderLabel.HorizontalAlignment = 'right';
            app.ScVarSliderLabel.Position = [761 422 47 15];
            app.ScVarSliderLabel.Text = 'Sc. Var.';

            % Create ScVarSlider
            app.ScVarSlider = uislider(app.UIFigure);
            app.ScVarSlider.Limits = [-3 3];
            app.ScVarSlider.ValueChangedFcn = createCallbackFcn(app,
                @ScVarSliderValueChanged, true);
            app.ScVarSlider.Position = [829 428 196 3];

```

```
            % Create CreategraphButton
            app.CreategraphButton = uibutton(app.UIFigure, 'push');
            app.CreategraphButton.ButtonPushedFcn = createCallbackFcn(app,
                @CreategraphButtonPushed, true);
            app.CreategraphButton.Position = [901 330 100 22];
            app.CreategraphButton.Text = 'Create graph';

            % Create LambdaEditField_2Label
            app.LambdaEditField_2Label = uilabel(app.UIFigure);
            app.LambdaEditField_2Label.HorizontalAlignment = 'right';
            app.LambdaEditField_2Label.Position = [663 137 50 15];
            app.LambdaEditField_2Label.Text = 'Lambda';

            % Create Lambda_m
            app.Lambda_m = uieditfield(app.UIFigure, 'numeric');
            app.Lambda_m.Position = [728 133 100 22];
            app.Lambda_m.Value = 0.9;

            % Create Lambda_M
            app.Lambda_M = uieditfield(app.UIFigure, 'numeric');
            app.Lambda_M.Position = [838 133 100 22];
            app.Lambda_M.Value = 1.1;

            % Create Beta_M
            app.Beta_M = uieditfield(app.UIFigure, 'numeric');
            app.Beta_M.Position = [838 100 100 22];
            app.Beta_M.Value = -0.2;

            % Create BetaEditField_2Label
            app.BetaEditField_2Label = uilabel(app.UIFigure);
            app.BetaEditField_2Label.HorizontalAlignment = 'right';
            app.BetaEditField_2Label.Position = [683 103 30 15];
            app.BetaEditField_2Label.Text = 'Beta';

            % Create Beta_m
            app.Beta_m = uieditfield(app.UIFigure, 'numeric');
            app.Beta_m.Position = [728 100 100 22];
            app.Beta_m.Value = -0.3;

            % Create Hs_M
            app.Hs_M = uieditfield(app.UIFigure, 'numeric');
            app.Hs_M.Position = [838 68 100 22];
            app.Hs_M.Value = 0.4;

            % Create HsEditField_2Label
            app.HsEditField_2Label = uilabel(app.UIFigure);
            app.HsEditField_2Label.HorizontalAlignment = 'right';
            app.HsEditField_2Label.Position = [688 70 25 15];
            app.HsEditField_2Label.Text = 'Hs';

```

```
            % Create Hs_m
            app.Hs_m = uieditfield(app.UIFigure, 'numeric');
            app.Hs_m.Position = [728 68 100 22];
            app.Hs_m.Value = 0.3;

            % Create Scalingvariable_M
            app.Scalingvariable_M = uieditfield(app.UIFigure, 'numeric');
            app.Scalingvariable_M.Position = [838 34 100 22];
            app.Scalingvariable_M.Value = 0.2;

            % Create ScalingvariablemMLabel
            app.ScalingvariablemMLabel = uilabel(app.UIFigure);
            app.ScalingvariablemMLabel.HorizontalAlignment = 'right';
            app.ScalingvariablemMLabel.Position = [588 37 125 15];
            app.ScalingvariablemMLabel.Text = 'Scaling variable offset';

            % Create Scalingvariable_m
            app.Scalingvariable_m = uieditfield(app.UIFigure, 'numeric');
            app.Scalingvariable_m.Position = [728 34 100 22];

            % Create MintestLabel
            app.MintestLabel = uilabel(app.UIFigure);
            app.MintestLabel.Position = [762 170 47 15];
            app.MintestLabel.Text = 'Min test';

            % Create MaxtestLabel
            app.MaxtestLabel = uilabel(app.UIFigure);
            app.MaxtestLabel.Position = [873 170 50 15];
            app.MaxtestLabel.Text = 'Max test';

            % Create lambda_min
            app.lambda_min = uieditfield(app.UIFigure, 'numeric');
            app.lambda_min.FontWeight = 'bold';
            app.lambda_min.FontAngle = 'italic';
            app.lambda_min.Position = [950 133 100 22];

            % Create beta_min
            app.beta_min = uieditfield(app.UIFigure, 'numeric');
            app.beta_min.FontWeight = 'bold';
            app.beta_min.FontAngle = 'italic';
            app.beta_min.Position = [950 100 100 22];

            % Create Hs_min
            app.Hs_min = uieditfield(app.UIFigure, 'numeric');
            app.Hs_min.FontWeight = 'bold';
            app.Hs_min.FontAngle = 'italic';
            app.Hs_min.Position = [950 68 100 22];

            % Create scaling_variable_min
            app.scaling_variable_min = uieditfield(app.UIFigure, 'numeric');
```

```
            app.scaling_variable_min.FontWeight = 'bold';
            app.scaling_variable_min.FontAngle = 'italic';
            app.scaling_variable_min.Position = [950 34 100 22];

            % Create MinErrorLabel
            app.MinErrorLabel = uilabel(app.UIFigure);
            app.MinErrorLabel.Position = [976 172 55 15];
            app.MinErrorLabel.Text = 'Min Error';

            % Create CalculateminimumButton
            app.CalculateminimumButton = uibutton(app.UIFigure, 'push');
            app.CalculateminimumButton.ButtonPushedFcn = createCallbackFcn(app,
                @CalculateminimumButtonPushed, true);
            app.CalculateminimumButton.Position = [828 203 120 22];
            app.CalculateminimumButton.Text = 'Calculate minimum';

            % Create LambdaEditField
            app.LambdaEditField = uieditfield(app.UIFigure, 'numeric');
            app.LambdaEditField.ValueChangedFcn = createCallbackFcn(app,
                @LambdaEditFieldValueChanged, true);
            app.LambdaEditField.Position = [1049 598 78 22];
            app.LambdaEditField.Value = 1;

            % Create BetaEditField
            app.BetaEditField = uieditfield(app.UIFigure, 'numeric');
            app.BetaEditField.ValueChangedFcn = createCallbackFcn(app,
                @BetaEditFieldValueChanged, true);
            app.BetaEditField.Position = [1049 531 78 22];

            % Create HsEditField
            app.HsEditField = uieditfield(app.UIFigure, 'numeric');
            app.HsEditField.ValueChangedFcn = createCallbackFcn(app,
                @HsEditFieldValueChanged, true);
            app.HsEditField.Position = [1049 467 78 22];
            app.HsEditField.Value = 0.3;

            % Create ScalingvariableoffsetEditField
            app.ScalingvariableoffsetEditField = uieditfield(app.UIFigure, '
                numeric');
            app.ScalingvariableoffsetEditField.Position = [1049 415 78 22];
        end
    end

    methods (Access = public)

        % Construct app
        function app = visualisation_scaling_AR()

            % Create and configure components
            createComponents(app)
```

```

            % Register the app with App Designer
            registerApp(app, app.UIFigure)

            % Execute the startup function
            runStartupFcn(app, @startupFcn)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```

Conclusion

Although the GUI was efficient to find approximately new quantum critical scalings for $Cu(C_4H_4N_2)(NO_3)_2$, $LiErF_4$ and $Ba_2CoGe_2O_7$, a lot can still be improved in the GUI. For example, one could find a way not to have to recompute completely the third order polynomial each time a parameter is changed. For the scaling of the dielectric constant of $Ba_2CoGe_2O_7$, one should define the QCR region more precisely and maybe have more data sets between 43 to 45 T and investigate why or why not there is a scaling for these magnetic fields.

References

[1] M.Jeong and H.M Rønnow, Phys. Rev. B **92**, 180409(R) (2015)

[2] M.Vojta, Rep. Prog. Phys. **66**, 2069 (2003)

[3] S. Sachdev *Quantum phase transitions*, 2nd ed. (Cambridge University Press, Cambridge, 2011).

[4] S. Sachdev and B. Keimer, Phys. Today **64** (2), 29 (2011)

[5] P. Babkevich, M. Jeong, Y. Matsumoto, I. Kovacevic, A. Finco, R. Toft-Peterson, C.Ritter, M.Månsson , S. Nakatsuji, and H.M. Rønnow, Phys. Rev. Lett. PLR **116**, 197202 (2016)

[6] G. Aeppli, A. Dutta, B.K. Chakrabarti, D. Sen, U. Divakaran, T.F. Rosenbaum, *Quantum Phase transitions in transverse field spin models : from statistical physics to quantum information*, (Cambridge University Press, Cambridge, 2015).

[7] Kim, J. W. *et al* Manifestation of magnetic quantum fluctuations in the dielectric properties. *Nat. Commun.* 5:4419 doi: 10.1038/ncomms5419 (2014)

[8] C. Kraemer, N. Nikseresht, J. O. Piatek, N. Tsyrulin, B.D. Piazza, K. Kiefer, B. Klemke, T.F. Rosenbaum, G. Aeppli, C. Gannarelli, K. Prokes, A. Podlesnyak, T. Strässle, A. Di Lieto, J. Jensen, and H.M Rønnow, Science **336**, 1416 (2012).