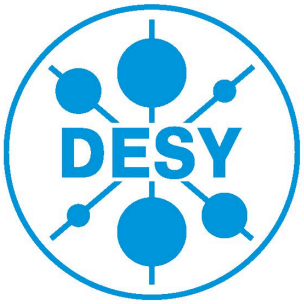


Graphical User Interface for Quench Spot Localisation

Benjamin Schröder

University of Göttingen, Germany

Supervisor: Felix Schlander (FLA)

Project period: July 19th - September 8th 2011

Abstract

This report describes my project at the DESY summer student programme in 2011. Mainly, the work contains the development of a graphical user interface (GUI) and a measuring device for a cavity mounted in a test insert.
Second sound, a quantum phenomenon in superfluid helium which behaves like a sound wave and transports heat, is used to localise spots on the cavity wall where a surface imperfection has induced a thermal breakdown of the superconducting cavity.
The GUI is used to read in sets of data of a second sound measurement, to process that data and finally, to display the determined spot where the breakdown started. Because of noise effects, a fully automated treatment is not efficient. Therefore, a user has to control the process.
A GUI has been created which embraces all requirements and is able to handle all types of used data. In addition a measurement device has been built. It allows a fast determination of the position of newly mounted cavities in the test insert.

Contents

1. Introduction

Current projects like the European X-Ray Free Electron Laser (XFEL) – which is being built at DESY – and the future International Linear Collider (ILC) require more powerful accelerator technologies. Superconducting cavities (resonators) are the most auspicious candidates for this purpose [1][2]. Thanks to the superconducting properties the impedance is orders of magnitudes smaller compared to normal conducting cavities and therefore, much higher energy gains are possible [3].
The focus lies on two physical quantities: The electrical field $\vec{E}$ which gives the energy gain of a charged particle accelerated by $\vec{E}$ and the quality factor Q which is defined as the total amount of stored energy divided by the energy loss per period. Currently applied nine-cell-cavities made of niobium normally do not reach field gradients more than 25 MV/m at Q-values up to some 10^{10}. Theoretical values of around 55 MV/m should be possible [3].
There are several reasons why these gradients are not reached so far. The most crucial effect is the local thermal breakdown of the superconductivity (*quench*): The applied cavities are driven by an alternating field with a frequency of 1.3 GHz. But due to surface imperfections, the cavities have spots with a higher impedance which causes Joule heating with increasing electromagnetic fields. When the temperature exceeds the critical temperature of the superconducting material – for niobium $T_C = 9.2$ K – the area around the quench spot turns into the normal conducting state. A huge ratio of the stored energy is released and the cavity cannot maintain the accelerating field. The cavity has to be cooled down by liquid helium to achieve superconductivity. Thus, a certain period is needed to cool down the quench spot again after a breakdown. That takes several ms. Most often breakdown producing imperfections are distributed on welding seams like the equator of a cell [3][4].

Localisation and inspection of the quench spots are important issues to improve the cavities. At DESY this research is done by the group of professor Eckhard Elsen. Two topics are treated by his Ph.D. students: On the one hand, a test frame was designed to use the so called *second sound* to find quench spots by triangulation. This report will describe the method of second sound driven localisation in more detail. On the other hand, a camera system to examine the located quench spots optically was developed [5].

The following chapter gives a brief overview of second sound and how to detect a quench position with it. Afterwards, the status of the project and the challenges – this report will deal with – are described. Finally, the main sections present the explanation of the developed solutions.

2. Quench Localisation - The Idea and Progress

2.1. Second Sound - Phenomenon and Detection

Helium shows remarkable properties when reaching temperatures below a critical value of $T_\lambda = 2.17$ K[1]: The heat conductivity increases by several orders of magnitude compared to the normal state and the viscosity for a temperature depending part of the helium is zero[2]. As it turned out, this state is a new phase which can be described by a two fluid model, first proposed by Laszlo Tisza and Lev Landau. Within their theory, entropy is carried by a sound like wave which is called *second sound*. The altering quantity in the wave equation is the ratio of a normal fluid density ρ_n and a superfluid density ρ_s while the total density $\rho = \rho_n + \rho_s$ is constant. This entropy wave is responsible for the fast heat flux. For a more detailed description compare [6] and [7]. Second sound has a temperature dependent velocity. In the temperature range within the cavities are used, the velocity lies between 16.8 m/s at 2 K and 19.9 m/s at 1.8 K [8].

One technical realisation of a second sound detector is a device shown in fig. 1. It is similar to a condenser microphone. The porous membrane is coated with a thin metal layer[3]; it represents one of the capacitor plates. The other electrode is made out of solid brass or an other conducting metal. By applying a high voltage the membrane is attracted.
When the second sound wave front arrives at the membrane the superfluid component is able to diffuse through the thin pores of the membrane while the normal fluid part is not able to. Due to the difference in the density of helium in- and outside the casing, diffusion occurs to compensate the density gradient. The distance d between the solid plate and the metal foil differs and so the capacitance C changes because of the distance dependency $C \sim {}^{A}/_{d}$ (plate area A is constant). This capacitance variation can be measured as a change in the voltage. The first peak gives the arrival time. The name of the device is *oscillating superleak transducer* (OST) [9].

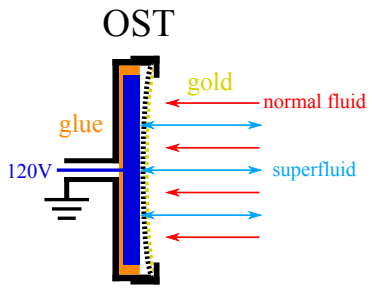


Figure 1 A sketch of an OST by Hannes Vennekate (University of Göttingen).

To detect a heat source in three dimensional space, at least three OSTs have to be used: The time of flight − time difference of the heat release and the first OST peak − gives the distance d_n between the OST n and heat source via the known velocity. For every OST a sphere with radius d_n is calculated. Finally, the intersection of these spheres marks the position of the source.

[1] Because of the shape of the specific heat curve, this temperature is also called λ-Point.
[2] There are much more unique properties but they will not be treated at this point.
[3] In this case Gold.

2.2. Status of the Project

The project involves the development and realisation of a cavity test stand to localise quench spots, based on the OST technique described above. The advantage of this method is the speed in comparison to other techniques like *temperature mapping*. At DESY (Hall III), four vertical test inserts already exist which can house the nine-cell-cavities; fig. 2 depicts one of them. The inserts are able to hold the OSTs as well. To gain sufficient data, eight microphones are mounted around the cavities at several heights.
A thermal breakdown is observed in a falling edge of the radio frequency (rf) performance. This peak gives the quench time. Some of the eight OSTs show a significant amplitude after a breakdown. By selecting those signals, the quench spot can be found via the triangulation described above. However, due to some geometrical constraints even two signal starts suffice for a localisation; but normally the found spot has a larger uncertainty.

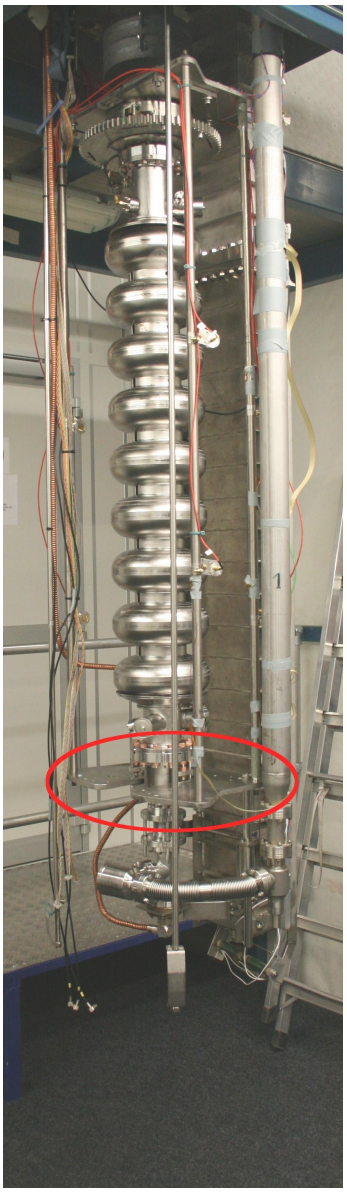


Figure 2 One of the four vertical test inserts. The red framed area indicates the cantilever (see next chapter).

During a cavity test two data sets are recorded for each resonator mode[4]. The first data set (in the following called `rf-` or `oo-data`) consists of the rf (performance) signal and the OST signals. For the second one (`worf-data`) the rf junction is disconnected. This procedure has to be done because it turned out that if the rf junction is connected, the OST signals sometimes are overlaid with noise. Particularly, the smaller OST amplitudes would not be resolvable anymore.
To make the two data sets comparable, the time scale of the second data has to be gauged. For that purpose the most intense OST signal start of the `oo-data` is selected − this can be executed by a programme or, in the case it failes, by hand − and then all OST signals of the `worf-data` will be shifted until both signal start times fit to each other. Now it is possible to get the times of flight for at least 2 OSTs. This, the number of chosen OSTs and the position of all OSTs are inserted in a programme which calculates the quench spot. This programme and the programme to compute the quench time have already been created by Felix Schlander. For a detailed description of the algorithm see [10].

[4]The modes reach from $\pi/9$ to π with $\pi/9$ increments.

2.3. Problems and Tasks

If a new cavity is mounted, it cannot be guaranteed that the position and arrangement stay the same as before. Therefore, the position of every recently installed cavity has to be metered relatively to the OSTs. This was so far done with a big expenditure of time. Furthermore, it was necessary to enter the data like the TOF and the chosen OSTs manually in the code. Felix Schlander has already begun to design a graphical user interface to solve the problems mentioned above.

The project can be subdivided in a design and a programming part. The design part consists of the development and construction of a device or of several devices for a fast cavity position metering.
The programming part covers the upgrade of the existing GUI: It should be flexible enough to handle the two kinds of data and it should offer the option to enter all necessary information over a simple and reliable interface.

3. Realisation

3.1. Measuring Device

The problem is treated in cylindrical coordinates (r, z, φ). It turned out − after an inspection of the mounting frame − that r varies insignificantly within a range of 2 mm. This value will be considered as an error σ_r. The resulting uncertainty of φ adds up to

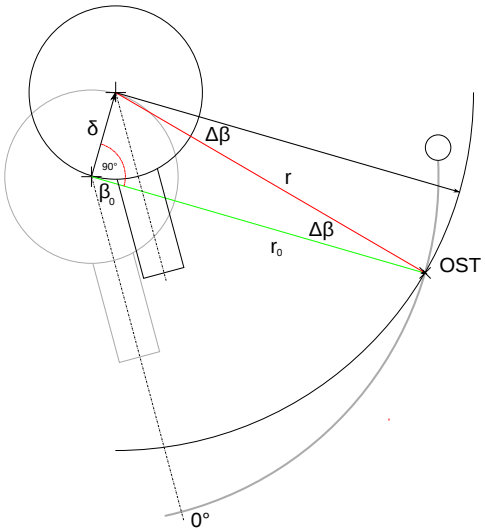


Figure 3 Geometry to estimate the error in the angle. The maximum is given by a shifting δ perpendicular to the radius r_0. The error is calculated via $\Delta\beta = \arctan(\delta/r_0)$. With typical values of r_0 of about 200 mm and a maximum shift of 2 mm, the angle uncertainty is almost 0.6°.

a maximum of about 0.6° for a shifting δ perpendicular to the original radius (fig. 3). Thus only z and φ have to be metered.
At the beginning, a direct measurement between a reference point on the cavity and the OSTs was considered. However, a better reference point is the test stand's cantilever (fig. 2): A solid and expanded metal plate which serves as a fixation for the cavity.

Every OST i has a fixed distance z_p^i to the plate and it is possible to attach an angle scale which defines the OST angle φ_r^i relative to the 90°-point of the scale[5].

Height Measurement The higher order mode (HOM) coupler (fig. 4 (left, red)) is chosen as cavity reference because of it's proximity to the cantilever. It is useful to define the centre of the fifth cell as the origin; 0° matches to the power coupler (fig. 5 (left, violet)). The measured difference z_m between the cantilever and the upper edge of the coupler can is metered by a marking gauge. Fig. 4 (right) illustrates such a measurement. The geometry is shown in fig. 4 (left). The prospected distance z^i is deduced by a simple calculation:

$$z^i = z_p^i - (z_r + z_m)$$

z_r is the distance between the power coupler and the centre of the fifth cell.

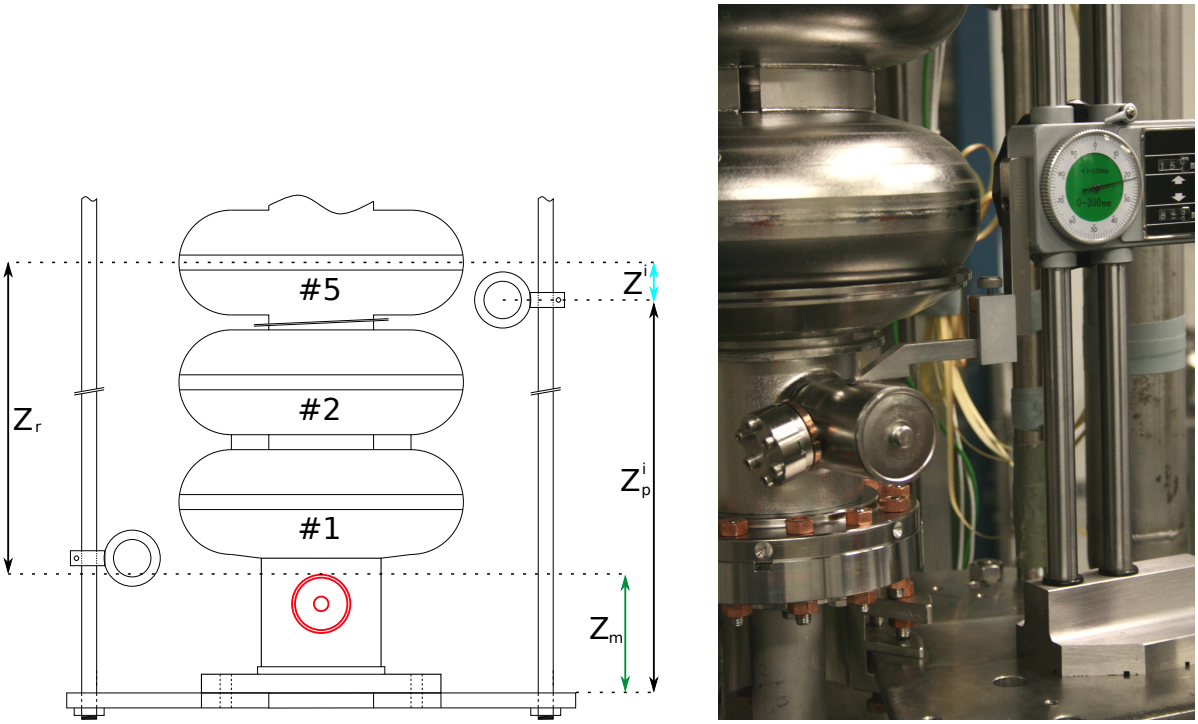


Figure 4 (left) Geometry of the height measurement. The necessary distance is z^i (turquoise); the metered one is z_m (green). All z_p^i positions for the OSTs have to be quantified one time only. z_r is defined by the cavity construction.

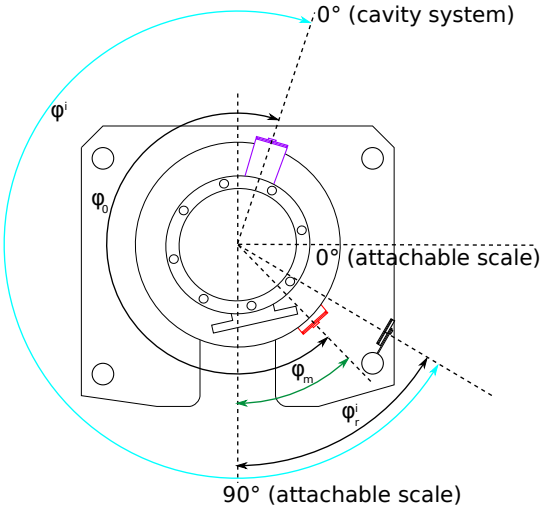
Angle Measurement An attachable plate with an angle scale was built for the φ_m measurement (fig. 5 (right)). A plumb bob is able to suspend from the HOM-coupler. The tip points to the scale. A blueprint with all proportions is given in fig. 8. Similar to

[5]Technically, it is easier to select 90° than 0° because of the scale arrangement on the plate.

the height, the angle φ^i between the middle of the OST i and the power coupler results from the geometry (fig. 5 (left)):

$$\varphi^i = \varphi_0 + (\varphi_r^i - \varphi_m) = \varphi_0 + ((90° - \varphi_r'^i) - (90° - \varphi_m')) = \varphi_0 + (\varphi_m' - \varphi_r'^i)$$

φ_0 is the angle between the HOM and the power coupler which is given by the cavity construction. The modification of 90° is necessary to rescale the measured values $\varphi_r'^i$ and φ_m' because of the reference point on the scale.



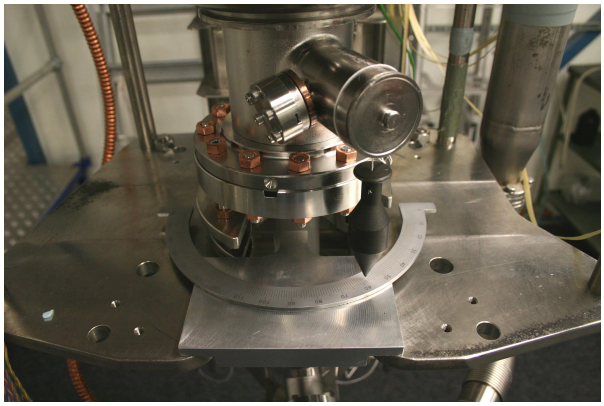


Figure 5 (left) Geometry of the angle measurement. The necessary angle is φ^i (turquoise); the metered one is φ_m (green). All φ_r^i positions for the OSTs have to be quantified one time only. φ_0 is defined by the cavity construction.

3.2. Graphical User Interface

All written and upgraded functions and routines can be found in the appendix. Table 1 and 2 list the folder and file structure with comments about the functions of the particular file. In Chapter A.2, the entire created functions are presented. The following chapter (A.3) tabulates the modifications of the already existing programmes (left column: modifications, right column: old code). `main.m` is the basic programme which is invoked by the GUI. Due to the major changes in the code, it is presented as one.
It is not necessary to read the functions for an understanding of the following chapters.

General Description The GUI was developed with `Matlab`. Fig. 6 illustrates the final interface. Two figures are used: In the left one the signals (OSTs and cavity performance) are plotted; in the right one a surface plot with a red dot − indicating the quench spot − will be drawn after the spot localisation has succeeded. In addition, the computed position of the quench spot will be quantified in the lower right area in cylindrical coordinates (`theta`, `rho` and `z`). The point of origin is again the centre of the fifth cavity cell.
A status bar on the bottom informs the user about errors which occured during the

process, proceedings and gives instructions about the further analysis.
For several tasks, data points of the signals have to be selected. This is achieved by applying the `Data cursor mode` which gives a cross mouse pointer. By clicking on a signal a movable datatip appears. It snaps automatically to the curves. If a new datatip is desired – to select the next signal start – the user can get one by a right click and then a left click on `Create New Datatip` on the appearing pop-up menu. If the resolution of the signal does not allow a good selection of start time, the `Zoom` button can be used to replot a smaller area.

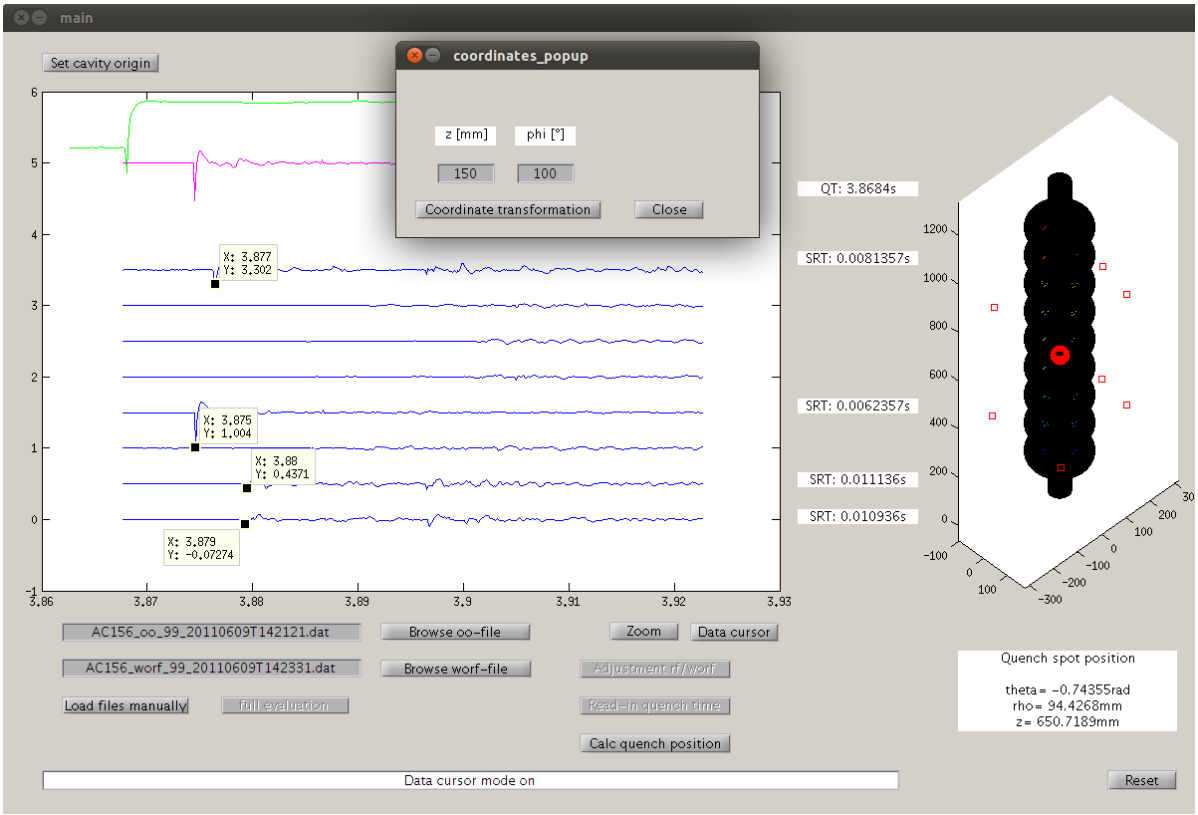


Figure 6 The developed graphical user interface. The eight blue curves are the OST signals of the data set without rf, the pink one is the OST signal which was used to gauge the two data sets and the green one is the rf performance progression. The negative peak at about 3.875 s indicates the quench time.

By clicking on `Set cavity origin` a new window appears (also shown in fig. 6) where it is possible to enter or check – in the case that those are already entered – the values φ_m and z_m to gauge the coordinate system.
`Reset` is used to remove all set variables and figures to restore the GUI for a new analysis. The two browser buttons are applied to read in the two data sets. By clicking on `Load files manually` the analysis will be started. All other buttons are explained in the next paragraph.

Procedure As mentioned before, there exist two kinds of data sets: For the first one the programme routine succeeds with the automatic determination of the quench time (in the code: `qtrf`). The user can directly select all identifiable start times and initiate the quench spot localisation by `Calc quench spot`. Between the two plot areas the quench time (`QT`) and signal running times (`SRT`) are displayed.

For the second kind this is not the case: The gauging has to be done by the user. If the programme cannot find the quench time, the OST signals of the first data set and the rf performance will be ploted over the entire measured period (usually 10 s). The user is supposed to select a reference start time. Logically, that one with the highest amplitude should be chosen. The OST number and selected time are read in by pressing `Adjustment rf/worf`.
Since the number of the OST is known, the programme can automatically find the start signal in the second data set, rearrange both time scales and plot the OST signals of the second data set in an adequate interval. Now the user has to select the quench time and to acquire it by clicking on `Read-in quench time`. The last step consists of the selection of all identifiable signal starts and the spot localisation (`Calc quench position`). The flow chart in fig. 7 summarises this procedure.

Verifications There are a lot of verifications implemented which shall avoid wrong inputs by the user. Some should be discussed here. By disabling buttons, it is guaranteed that no unscheduled analysis step will be applied. For example, if the programme was able to find the quench time automatically, it is impossible to execute an adjustment or to read in a quench time by hand. An other example is that any analysis step cannot be achieved until data sets are read in.
Furthermore, kinds of signals are compared. For instance, if the user is supposed to select a time on the rf signal − but the selected point is one of an OST signal − the procedure is interrupted and the user is requested to choose the right curve by an instruction on the status bar.
In case of an error, which cannot be handled by the GUI, it is advisable to reset (`Reset` button) the system and start again with the analysis.

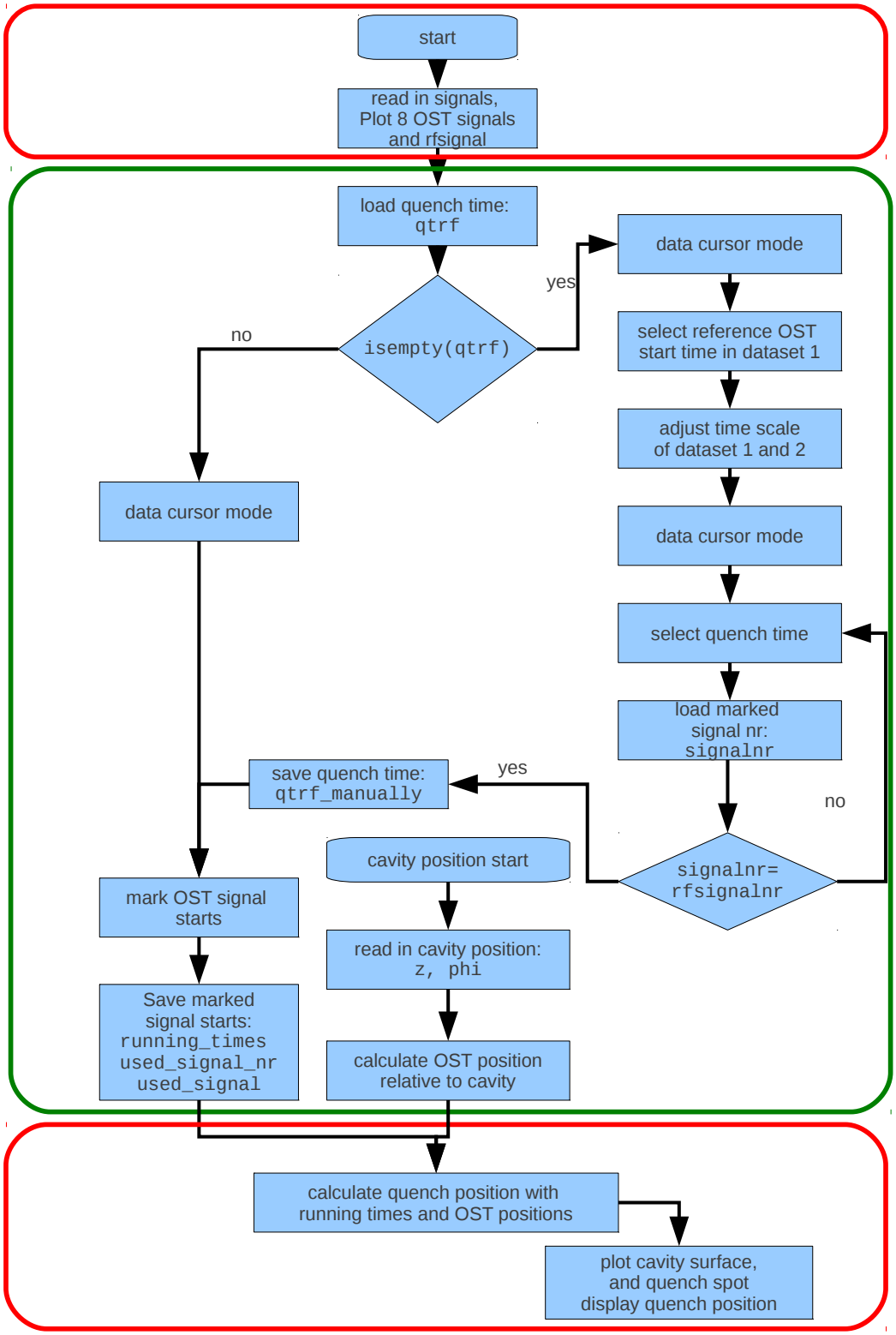


Figure 7 Flow chart of the procedure: the red framed areas were already written by Felix. This parts are simplified and many intermediate steps are neglected. The green framed is the part of this project.

4. Conclusion

During the period of the summer student programme, it was possible to become acquainted with the `Matlab` environment to generate graphical user interfaces. Because of previous knowledge, it took only a few days to get familiar with the other theoretical knowledge about the second sound topic and about Felix's project.
We considered a lot of ideas concerning the gauging devices and many sketches were drawn until we got that simple solution described above. It was easy to build and only two coordinates have to be measured. Previously, it took almost half an hour to meter all eight OST positions relative to the centre of the cavity. With the new idea it only needs two minutes.
The upgrade of the GUI gives control of the two types of data and allows the user to select all identifiable signal start times manually. This replaces a routine which determines the start times automatically. Such a routine would be unreliable due to the noise and highly varying amplitudes. The GUI combines the programmes written by Felix Schlander in an easy operable terminal. The user is able to review each analysis step in the plot areas and the status bar.

Acknowledgement I would like to thank my supervisor Felix Schlander and professor Elsen for the opportunity to work with the FLA group, for all the help I got and for my exciting time at DESY. Also thankworthy is the work of Uwe Cornett who contributed many ideas and built the angle scale plate. Last but not least, I want to thank Olaf Behnke, Andrea Schrader and their team for the great organisation of the summer student programme.

A. GUI

A.1. Folder and File Structure

Created files	Function
`adjustment.m`	Rearranges time scales
`calc_quench_spot.m`	Invokes quench localisation
`coordinates_popup.fig`	GUI for pop-up menu
`coordinates_popup.m`	Handle file for pop-up menu
`coordinate_transformation.m`	Calculates OST positon relative to cavity
`plot_rfsignals.m`	Plots data set one and rf power
`runningtime.m`	Calculates time of flight

Modified files	Function
`dataevaluation.m`	Processes and plots data
`main.m`	Handle file for the main GUI
`openfiles.m`	Main file for quench time calculation
`findspot.m`	Calculates the quench spot
`main_quenchpos.m`	Invokes quench localisation
`setdetectors.m`	Set detector positions relative to cavity

Table 1 List of created and modified code.

Type	Original name	New name
folder	`origquenchlocmitwinkelmitcavity2/`	`quench_loc_2OST`
folder	`origquenchlocmitwinkelmitcavity3/`	`quench_loc_3OST`
folder	`origquenchlocmitwinkelmitcavity4/`	`quench_loc_4OST`
file	`./quench_loc_2OST/main.m`	`./quench_loc_2OST/main_quenchpos.m`
file	`./quench_loc_3OST/main.m`	`./quench_loc_3OST/main_quenchpos.m`
file	`./quench_loc_4OST/main.m`	`./quench_loc_4OST/main_quenchpos.m`

Table 2 List of modified file and folder names.

A.2. Created Code

main.m

```
[...]

% --- Executes on button press in btn_files.
function btn_files_Callback(hObject, eventdata, handles)
% hObject    handle to btn_files (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.edt_status, 'String', 'Loading...', 'BackgroundColor', [1 1 1]);
pause(0.1);     % needed to update 'handles.edt_status'
setappdata(0, 'handles', handles);      % will be loaded in 'dataevaluation.m'
oofile=get(handles.edt_filerf,'String');
worffile=get(handles.edt_fileworf,'String');

% verification if a rf AND a worf file are filled in
if strcmp(oofile, 'file') || strcmp(worffile, 'file')
    set(handles.edt_status, 'String', 'No files browsed', 'BackgroundColor', 'red');
else
    data=openfiles(oofile, worffile);
    %check whether a quench time is automatically found or if a quench time
    %has to be picked manually; 'data' is empty in the first case
    if isempty(data)
        plot_rfsignals(handles, oofile, worffile)
        set(handles.adjustment_rf_worf, 'Enable', 'on')
        set(handles.edt_status, 'String', 'Loaded. No quench time detected. Select OST reference!',...
            'BackgroundColor', 'yellow');
    else
        set(handles.calc_quench_position, 'Enable', 'on')
        set(handles.edt_status, 'String',...
            'Loaded. Quench time detected. Select OST signal starts!', 'BackgroundColor', 'green');
    end
end
guidata(hObject, handles);  % store GUI data for further processing

[...]

rfsignalnr=getappdata(0, 'rfsignalnr');     % retrieve plot handle of rf plot; is stored in 'dataevaluation.m'
% verification that 'rfsignalnr' is not empty; means that a the plots are already done
if isempty(rfsignalnr)~=1
    cursor=datacursormode(gcf);
    set(cursor,'Enable', 'on', 'SnapToDataVertex', 'off');
    setappdata(0, 'cursor', cursor); % store cursor handle
    set(handles.edt_status, 'String', 'Data cursor mode on', 'BackgroundColor', [1 1 1])
else
    set(handles.edt_status, 'String', 'No signals are loaded', 'BackgroundColor', 'red')
end


% --- Executes on button press in zoom.
function zoom_Callback(hObject, eventdata, handles)
% hObject    handle to zoom (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

zoom(handles.axes1, 'on')
set(handles.edt_status, 'String', 'Zoom mode on', 'BackgroundColor', [1 1 1])


% --- Executes on button press in adjustment_rf_worf.
function adjustment_rf_worf_Callback(hObject, eventdata, handles)
% hObject    handle to adjustment_rf_worf (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

cursor=getappdata(0, 'cursor');
cursor_info=getCursorInfo(cursor);

if isempty(cursor_info)~=1
    signalnr=getappdata(0, 'signalnr');
```

```
    rfsignalnr=getappdata(0, 'rfsignalnr');

    if cursor_info(1).Target ~= rfsignalnr
        [~, ref_ost_info(1)] = max(signalnr == cursor_info.Target);
        ref_ost_info(2)=cursor_info.Position(1);
        adjustment(handles, ref_ost_info)
        set(handles.adjustment_rf_worf, 'Enable', 'off')
        set(handles.read_in_quench_time, 'Enable', 'on')
        set(handles.edt_status, 'String', ...
            'Reference OST start picked. RF and WORF data adjusted. Select quench time!',...
            'BackgroundColor', 'green')
    else
        set(handles.edt_status, 'String', 'Pick a OST start instead of rf signal', ...
        'BackgroundColor', 'red')
    end
else
    set(handles.edt_status, 'String', 'No reference OST start signal selected', ...
        'BackgroundColor', 'red')
end


% --- Executes on button press in read_in_quench_time.
function read_in_quench_time_Callback(hObject, eventdata, handles)
% hObject    handle to read_in_quench_time (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

cursor=getappdata(0, 'cursor');
cursor_info=getCursorInfo(cursor);
rfsignalnr=getappdata(0, 'rfsignalnr');

% verification that 'cursor_info' is not empty; means that a cursor tip exists
if isempty(cursor_info)~=1
    % verification that picked data point is an acual rfsignal data point
    if cursor_info.Target == rfsignalnr
        qtrf_manually=cursor_info.Position(1);
        setappdata(0, 'qtrf', qtrf_manually)
        set(handles.read_in_quench_time, 'Enable', 'off')
        set(handles.calc_quench_position, 'Enable', 'on')
        set(handles.edt_status, 'String', ...
            'Quench time manually picked. Select OST signal starts!', 'BackgroundColor', 'green')
    else
        set(handles.edt_status, 'String', 'No or wrong signal selected. Select quench time!',
'BackgroundColor', 'red')
    end
end


% --- Executes on button press in calc_quench_position.
function calc_quench_position_Callback(hObject, eventdata, handles)
% hObject    handle to calc_quench_position (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

cursor=getappdata(0, 'cursor');

if isempty(cursor)~=1
    cursor_info=getCursorInfo(cursor);
    if size(cursor_info,2) > 1
        datacursormode off
        set(gcf, 'CurrentAxes', handles.axes2)
        [running_times used_signals used_signal_nr]=runningtime(hObject, eventdata, handles)
        calc_quench_spot(hObject, eventdata, handles, running_times, used_signals, used_signal_nr)
        view(45,45)
    else
        set(handles.edt_status, 'BackgroundColor', 'red', ...
            'String', 'Only one signal start picked. At least 2 are needed!')
    end
else
    set(handles.edt_status, 'BackgroundColor', 'red', 'String', 'Missing signal start marks')
end

```

```
% --- Executes on button press in set_cavity_origin.
function set_cavity_origin_Callback(hObject, eventdata, handles)
% hObject    handle to set_cavity_origin (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

coordinates_popup

% --- Executes on button press in reset.
function reset_Callback(hObject, eventdata, handles)
% hObject    handle to reset (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

cla(handles.axes1)
cla(handles.axes2)

setappdata(0, 'cursor', [])
setappdata(0, 'signalnr', [])
setappdata(0, 'rfsignalnr', [])

set(handles.edt_filerf,'String', 'file')
set(handles.edt_fileworf,'String', 'file')

set(handles.adjustment_rf_worf, 'Enable', 'off')
set(handles.read_in_quench_time, 'Enable', 'off')
set(handles.calc_quench_position, 'Enable', 'off')

datacursormode off
set(gcf, 'CurrentAxes', handles.axes1)
delete(findall(gca, 'Type','hggroup','HandleVisibility','off','Draggable','on'))

hold(handles.axes1, 'off')
hold(handles.axes2, 'off')

set(handles.edt_status, 'String', 'All variables reseted. New measurement possible.', ...
    'BackgroundColor', [1 1 1])

set(handles.quench_time, 'String', 'Quench time')
set(handles.quench_pos, 'String', 'Quench spot position')

for j=1:8
     set(eval(['handles.OST' num2str(j) '_running_time']), 'String',...
         ['OST' num2str(j) '_run_time'], 'BackgroundColor', [1 1 1])
end
```

adjustment.m

```
function adjustment(handles, ref_ost_info)
% function: fits the time scale of the second data set to the first by
% shifting the time scale unitl start time fits to ref_ost_info

threshold_sigma = 5;                    % mulitplicator: defines the threshold
rfdata=getappdata(0, 'rfdata');
worfdata=getappdata(0, 'worfdata');
dt=getappdata(0, 'dt');                 % sampling rate
pl_rg = 500;                            % plot range

ref_ost_idx = ref_ost_info(2)/dt;

% average of the signal
average_worf = sum(worfdata(:, ref_ost_info(1)+2)) / length(worfdata(:, ref_ost_info(1)+2));
% corrected signal
cor_worfdata = worfdata(:, ref_ost_info(1)+2) - average_worf;
threshold = threshold_sigma*sum(abs(cor_worfdata)) / length(cor_worfdata);

% select two signal start indexes
[~, signal_start_idx_th] = max(abs(cor_worfdata) > threshold)
[~, signal_start_idx_min]= min(cor_worfdata)

% compare the two indexes if they do not fit within a interval of 20 stop
% the program
if (signal_start_idx_min < signal_start_idx_th - 10) || (signal_start_idx_min >
signal_start_idx_th + 10)
    set(handles.edt_status, 'String', 'Could not find signal start in picked reference OST.
Process aborted.')
    return
end

% plot the rf signal
average = sum(rfdata(:,2))/length(rfdata(:,2));
rfsignalnr=plot(handles.axes1, rfdata(ref_ost_idx-pl_rg:ref_ost_idx+pl_rg,1), ...
    rfdata(ref_ost_idx-pl_rg:ref_ost_idx+pl_rg,2)-average+5.5, 'g');

hold(handles.axes1, 'on')

% plot the chosen OST signal of the first data set
average = sum(rfdata(:,ref_ost_info(1)+2))/length(rfdata(:,ref_ost_info(1)+2));
plot(handles.axes1, rfdata(ref_ost_idx-pl_rg:ref_ost_idx+pl_rg,1), ...
    rfdata(ref_ost_idx-pl_rg:ref_ost_idx+pl_rg,ref_ost_info(1)+2)-average+5, 'm');

% calc the index shift to gauge the two data sets
shift=signal_start_idx_min-ref_ost_idx

% plot the 8 OST signals of the shifted second data set
for i=1:8
    average = sum(worfdata(:,i+2)) / length(worfdata(:,i+2));
    shifted_data = circshift(worfdata(:,i+2), -shift);
    signalnr(i) = plot(handles.axes1, worfdata(ref_ost_idx-pl_rg:ref_ost_idx+pl_rg,1), ...
        shifted_data(ref_ost_idx-pl_rg:ref_ost_idx+pl_rg)-average+(i-1)*0.5);
end

% save the plot handles
setappdata(0, 'signalnr', signalnr)
setappdata(0, 'rfsignalnr', rfsignalnr)
hold(handles.axes1, 'off')
```

calc_quench_spot.m

```
function calc_quench_spot(hObject, eventdata, handles, running_times, used_signals,
used_signal_nr)
% function: reads in the number of selected OST start times and invokes the
% particular program

pause(1)    % for one second the number of chosen OSTs will be displayed in the status bar
set(handles.edt_status, 'String', 'Calculating quench spot position...', 'BackgroundColor', [1 1
1])
pause(0.1)  % is needed to update the status bar
num_used_signals=sum(used_signals);

% switch distinguishes between the 3 cases, sets the needed path, invokes
% the program and afterwards removes the path again
switch num_used_signals
    case 2
        path(path, [pwd '/quench_loc_2OST']);
        main_quenchpos(handles, running_times, used_signal_nr)
        rmpath([pwd '/quench_loc_2OST'])
    case 3
        path(path, [pwd '/quench_loc_3OST']);
        main_quenchpos(handles, running_times, used_signal_nr)
        rmpath([pwd '/quench_loc_3OST'])
    case 4
        path(path, [pwd '/quench_loc_4OST']);
        main_quenchpos(handles, running_times, used_signal_nr)
        rmpath([pwd '/quench_loc_4OST'])
end

set(handles.edt_status, 'String', 'Quench spot position calculated', 'BackgroundColor', 'green')
```

coord_transformation.m

```
function [phi z] = coord_transformation(phi_m, z_m)
% function: calculates the position of the 8 OSTs relative to the cavity
% origin in cylindrical coordinates; the cav origin is set to the center of
% the fifth cell
% output: 2 8x1 vectors for phi and z; every entry is for one OST

phi_0 = 245;          % counter clock wise angle between hom and powercoupler
z_r = 541.6;          % dinstance between the center of the fifth cell and the upper edge of the
hom coupler

% phi and z values of the OSTs relative to the reference point 90°
phi_r = [0; 0; 0; 0; 0; 0; 0; 0];
z_p   = [0; 0; 0; 0; 0; 0; 0; 0];

% transformation; for more detail see report
phi = phi_r + phi_0 - phi_m - 180
z   = z_p - z_m - z_r
```

```
[...]

% --- Executes just before coordinates_popup is made visible.
function coordinates_popup_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to coordinates_popup (see VARARGIN)

% Choose default command line output for coordinates_popup
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% read out currently used z and phi coordinates of the cavity
set(handles.z, 'String', getappdata(0, 'meas_cav_coord_z'))
set(handles.phi, 'String', getappdata(0, 'meas_cav_coord_phi'))

% UIWAIT makes coordinates_popup wait for user response (see UIRESUME)
% uiwait(handles.figure1);

[...]

% --- Executes on button press in coordinate_transformation.
function coordinate_transformation_Callback(hObject, eventdata, handles)
% hObject    handle to coordinate_transformation (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

phi=str2num(get(handles.phi, 'String'));
z=str2num(get(handles.z, 'String'));

setappdata(0, 'meas_cav_coord_phi', phi)
setappdata(0, 'meas_cav_coord_z', z)

% invoke function which calculates the coordination transformation
[phi_prime z_prime] = coord_transformation(phi, z);

setappdata(0, 'trans_ost_coord_phi', phi_prime)
setappdata(0, 'trans_ost_coord_z', z_prime)

[...]

% --- Executes on button press in close.
function close_Callback(hObject, eventdata, handles)
% hObject    handle to close (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

close(handles.figure1)  % closes the figure
```

runningtime.m

```
function [running_times used_signals used_signal_nr]=runningtime(hObject, eventdata, handles)
% function: read in picked OST signals and manually or automatically picked Quenchtime;
% write running time for each picked OST and write quench time
% output: vectors running_times (8 by 1), used_signals (8 by 1) and used_signals_nr (#picked OSTs
by 1)

cursor=getappdata(0,'cursor');      % load the cursor handle
signalnr=getappdata(0,'signalnr');  % load the plot handles
cursor_info=getCursorInfo(cursor);  % read out cursor information: position, targets (plots) of
the datatips
num_datatips=size(cursor_info,2);   % number of choosen datatips
qtrf=getappdata(0, 'qtrf');         % quenchtime from RF data; was stored in 'openfiles.m' or
manually in 'main.m'
used_signals=zeros(8,1);            % vector with '1' if OST Signal is choosen and '0' if not
runnig_times=zeros(8,1);            % vector with determined running times for each OST
used_signal_nr=[];

% find choosen OSTs by passing the datatips and write running time
for i=1:num_datatips
    temp=signalnr-cursor_info(i).Target;
    for j=1:8
        if(temp(j)==0)
            used_signals(j)=1;
            used_signal_nr=[used_signal_nr; j];
            running_times(j,:)=cursor_info(i).Position(1)-qtrf;
            set(eval(['handles.OST' num2str(j) '_running_time']), 'String',...
                ['SRT: ' num2str(running_times(j)) 's'], 'BackgroundColor', [1 1 1])
        end
    end
end

% Fade out not used running time text arrays
for i=1:8
    if(used_signals(i)==0)
        set(eval(['handles.OST' num2str(i) '_running_time']), 'String', '',...
            'BackgroundColor', [0.831 0.816 0.784])
    end
end

set(handles.edt_status, 'String', [num2str(num_datatips) ' OST signals are selected'], ...
    'BackgroundColor', 'green')
set(handles.quench_time, 'String', ['QT: ' num2str(qtrf) 's'], ...
    'BackgroundColor', [1 1 1])
```

plot_rfsignals.m

```
function plot_rf_signals(handles, oofile, worffile)
% function: plots rf signals in case of no automatically picked quench time
% over the entire time scale; otherwise that is done by 'dataevaluation.m'

rfdata=dlmread(oofile, ' ', 2, 0);
worfdata=dlmread(worffile, ' ', 2, 0);
setappdata(0, 'rfdata', rfdata);
setappdata(0, 'worfdata', worfdata);

% plot the rf signal without offset
average = sum(rfdata(:,2))/length(rfdata(:,2));
rfsignalnr=plot(handles.axes1, rfdata(:,1), rfdata(:,2)-average+5, 'g');
hold(handles.axes1, 'on')

% plot the 8 OST signals of the first data set
for i=1:8
    average = sum(rfdata(:,i+2)) / length(rfdata(:,i+2));
    signalnr(i) = plot(handles.axes1, rfdata(:,1), rfdata(:,i+2)-average+(i-1)*0.5, 'm');
end

% save the plot handles to compare them later
setappdata(0, 'signalnr', signalnr)
setappdata(0, 'rfsignalnr', rfsignalnr)
hold(handles.axes1, 'off')
```

A.3. Modified Code

dataevaluation.m

openfiles.m


```
function data = openfiles(file1,file2)
%OPENFILES Summary of this function goes here
%  Detailed explanation goes here
tic
i=1;
ticklength=0.0001;

setappdata(0, 'dt', ticklength)     % store ticklength; needed in 'adjustment

[t1,rf1,s1]=calldata(file1);
[t2,rf2,s2]=calldata(file2);

%qtrf is cell of structure {quenchtime, [a b c d]} while quenchtime is
%the calculated quenchtime and [a b c d] are the fitting parameters
qtrf=quenchtime(t1,rf1);
dataset(i,1)={file1};
if qtrf{1}==-1
 ['no quench has been detected in ' file1 ' - please check manually']
 data=[];
 return
else
   qts=num2str(qtrf{1});
   [t1,rf1,s1,s1q,s2]=preparedata(t1,rf1,s1,qtrf{1},s2);
   ['t=' qts ' in ' file1]
dataset(:,2)={t1(floor(qtrf{1}*10000)-55:floor(qtrf{1}*10000)+545,:)};
dataset(:,3)={rf1(floor(qtrf{1}*10000)-55:floor(qtrf{1}*10000)+545,:)};
   dataset(:,4)={s1q};

%signals is a 3x10 cell array, first 8 rows contain "rf-free" signals
%,the fitting parameters ( {ostsignal, [a b c d]}) and the calculated propaga
%Row 9 contains
%the reference OST number and the corresponding propagation time. Row
%10 Col 1 is the corresponding dataset of the reference OST
signals=proptimes(ticklength,qtrf{1},dataset{i,4},s2,s1q);
end
data=dataevaluation(qtrf,{dataset{i,2},dataset{i,3}},signals,ticklength);

setappdata(0, 'qtrf', qtrf{1})      % store quench time globally; needed in '

toc
end
```

findspots.m

main_quenchpos.m

```
function main_quenchpos(handles, running_times, used_signal_nr)
setdetectors
teslashape

%Set detectors with a signal and the corresponding TOF tracecalc(detno,TOF
%in ms, detno, TOF in ms)
%quenchloc(3,5,6)
%tracecalc(1,18.064,2,20.809,5,18.93)
findspot(used_signal_nr(1),running_times(used_signal_nr(1)),...
    used_signal_nr(2),running_times(used_signal_nr(2)), handles)
```

main.m

```
function main
setdetectors
teslashape

%Set detectors with a signal and the corresponding TOF tracecalc(detno,TOF
%in ms, detno, TOF in ms)
%quenchloc(3,5,6)
%tracecalc(1,18.064,2,20.809,5,18.93)
findspot(1,10.8,7,12.5)
```

setdetectors.m

```
function setdetectors
% enter detector positioning data
% PLEASE NOTE: The coordinates of the detectors have to be given related to
% the equator of cell 5 in cylindrical coordinates. The angle is measured
% counterclockwise related to the coupling tube and up to 360 degrees.
setappdata(0,'detectorposition',[]);
% set detector data (radius reduced by equatorial cav.  radius,angle,z)
% will be recalculated in function "detectors"

phi = getappdata(0, 'trans ost coord phi');
z   = getappdata(0, 'trans ost coord z');

% detectors(96,     phi(1), z(1))
% detectors(96,     phi(2), z(2))
% detectors(119,    phi(3), z(3))
% detectors(112,    phi(4), z(4))
% detectors(111,    phi(5), z(5))
% detectors(113,    phi(6), z(6))
% detectors(102,    phi(7), z(7))
% detectors(102,    phi(8), z(8))

detectors(96,317,-334.2)
detectors(96,317,106.4)
detectors(119,253,-216.8)
detectors(112,253,232.8)
detectors(111,99,-341.2)
detectors(113,99,123.4)
detectors(102,31,-223.8)
detectors(102,31,231.8)
detpos=getappdata(0,'detectorposition');
size(detpos)
hold on

% plot detectors
for i=1:1:size(detpos,1)
plot3(detpos(i,1),detpos(i,2),detpos(i,3),'sr')
end
```

setdetectors.m

```
function setdetectors
% enter detector positioning data
% PLEASE NOTE: The coordinates of the detectors have to be given related to
% the equator of cell 5 in cylindrical coordinates. The angle is measured
% counterclockwise related to the coupling tube and up to 360 degrees.
setappdata(0,'detectorposition',[]);
% set detector data (radius reduced by equatorial cav.  radius,angle,z)
% will be recalculated in function "detectors"
detectors(96,317,-334.2)
detectors(96,317,106.4)
detectors(119,253,-216.8)
detectors(112,253,232.8)
detectors(111,99,-341.2)
detectors(113,99,123.4)
detectors(102,31,-223.8)
detectors(102,31,231.8)
detpos=getappdata(0,'detectorposition');
size(detpos)
hold on

% plot detectors
for i=1:1:size(detpos,1)
plot3(detpos(i,1),detpos(i,2),detpos(i,3),'sr')
end
```

B. Sketches

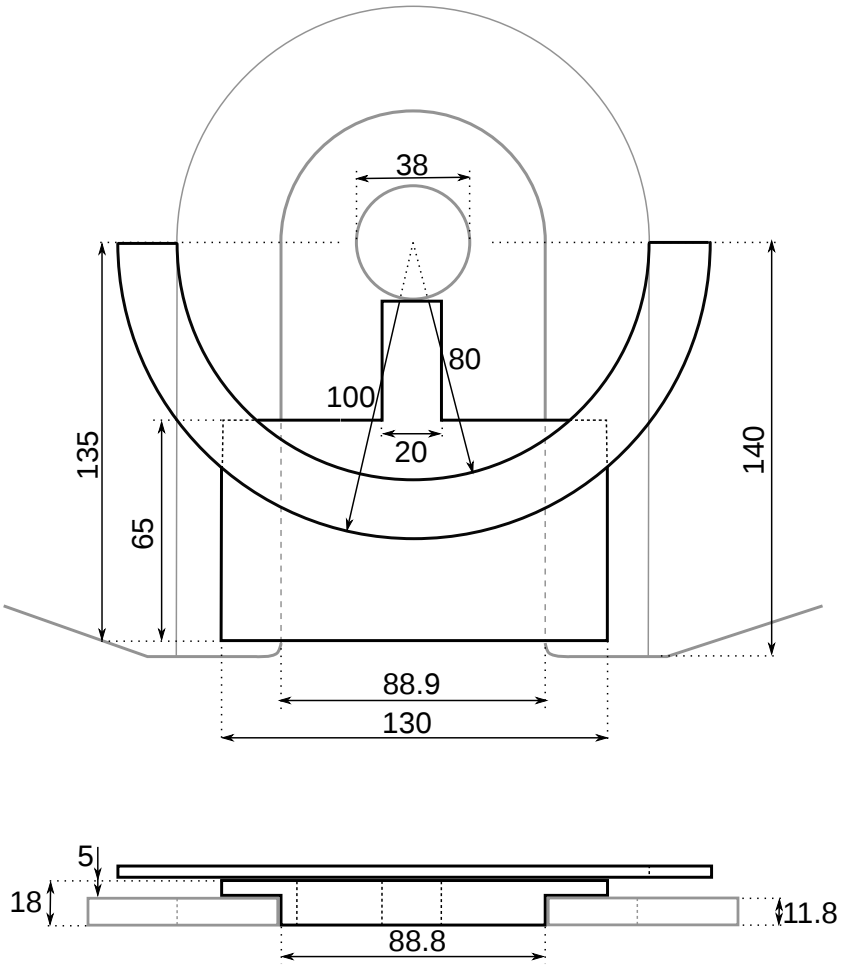


Figure 8 Blueprint of the plate with angle scale. All distances are given in mm. The cantilever is illustrated in grey.

C. References

[1] XFEL Technical Design Report; DESY 2006-097, Hamburg, July 2007

[2] ILC Technical Design Report; ILC-Report-2007-001, August 2007

[3] *H. Padamsee, J. Knoblauf, T. Hays*: RF Superconductivity for Accelerators; Second Edition, Weinheim, Wiley-VCH Verlag, 2008

[4] *J. Norem und M. Pellin*: Gradient limits and SCRF performance; Argonne, USA

[5] *F. Schlander, S. Aderhold, E. Elsen and D. Reschke*: Progress on Diagnostic Tools for Superconducting High-gradient Cavities; Proceedings of Linear Accelerator Conference LINAC2010, Tsukuba, Japan, THP014, 791-793, 2010

[6] *R. J. Donelly*: The two-fluid theory and second sound in liquid helium; Phys. Today, 34-39, Oct. 2009

[7] *L. Tisza*: Transport Phenomena in Helium II; Nature, **141**, 913, 1938

[8] *R. T. Wang, W. T. Wagner, R. J. Donnelly*: Precision Second-Sound Velocity Measurements in Helium II; J. Low Temp. Phys. **68**, 409-417, 1987

[9] *R. A. Sherlock and D. O. Edwards*: Oscillating Superleak Second Sound Transducers; Rev. of Scientific Instruments, **41**, 1603, 1970

[10] *F. Schlander and E. Elsen*: 2nd Sound as quench localisation tool; ILC-HiGrade-Report-2010-0071, 2010